BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2017 Bc. Balázs Šidó

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

TRACKING THE MOVEMENT OF OBJECTS IN THE VIDEO SIGNAL
SLEDOVÁNÍ POHYBU OBJEKTŮ V OBRAZOVÉM SIGNÁLU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Balázs Šidó**
AUTOR PRÁCE

SUPERVISOR — **Ing. Jiří Přinosil, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Balázs Šidó ***ID:*** 154885
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Sledování pohybu objektů v obrazovém signálu

POKYNY PRO VYPRACOVÁNÍ:

Proveďte analýzu stávajících metod pro sledování trajektorie jednotlivých pohybujících se objektů na základě znalosti pozic všech objektů v daném systému v daném čase. Vytipujte metody vhodné pro modelování pohybu proměnlivého počtu objektů a uveďte jejich případná omezení. Na základě zjištěných poznatků proveďte návrh a realizaci vlastní metody pro sledování trajektorie proměnlivého počtu objektů. Realizovanou metodu otestujte na databázi 2D MOT 2015.

DOPORUČENÁ LITERATURA:

[1] BENFOLD, Ben; REID, Ian. Stable multi-target tracking in real-time surveillance video. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011. p. 3457-3464.

[2] VO, Ba-Ngu, et al. Joint detection and estimation of multiple objects from image observations. IEEE Transactions on Signal Processing, 2010, 58.10: 5129-5141.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Jiří Přinosil, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRACT

This thesis focuses on multiple object tracking in clutter environment. The paper describes two implementations of filters in detail, that are essentially based on the Kalman Filter. Both implementations are based on tracking multiple objects with the knowledge of their positions in every frame. The first implementation is a hybrid of the Global and the Standard Nearest Neighbour Filters. The second implementation takes on a probabilistic approach to the data association process. The last chapter provides a comparison between these trackers and also between the Basic filter. The trackers were implemented in java.

KEYWORDS

Multiple target tracking, Kalman Filter, Nearest-Neighbour Filter, Probabilistic Data Association

ABSTRAKT

Tato diplomová práce se zaměřuje na sledování pohybu více objektů. Práce popisuje dvě implementace filtrů, které jsou v podstatě založeny na principu Kalmanova filtru. Obě implementace jsou založeny na principu sledování více objektů, na základě znalosti pozic všech objektů v každém snímku. První implementace je smíšená verze Globálního a Standardního filtru nejbližších sousedů. Druhá implementace je postavena na pravděpodobnostním přístupu k procesu sdružení. Poslední kapitola poskytuje srovnání mezi těmito filtry a Základním filtrem. Algoritmy byly realizovány v javě.

KLÍČOVÁ SLOVA

Sledování pohybu více objektů, Kalmanův filtr, Filtr nejbližších sousedů, Pravděpodobnostní sdružování dat

ŠIDÓ, Balázs *Tracking the movement of objects in the video signal*: master's thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Ústav telekomunikací, 2016. 62 p. Supervised by Ing. Jiří Přinosil, Ph.D.

DECLARATION

I declare that I have written my master's thesis on the theme of "Tracking the movement of objects in the video signal" independently, under the guidance of the master's thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the master's thesis I furthermore declare that, as regards the creation of this master's thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Sb., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No 40/2009 Sb., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would like to thank my supervisor, Ing. Jiří Přinosil, Ph.D. for his help, valuable advice and time during the work on this thesis.

Brno . . . .

author's signature

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

BRNO ....

podpis autora

CONTENTS

LIST OF FIGURES

INTRODUCTION

To track an object is to estimate it's true location, path and other characteristics from some given measurements. These measurements are acquired from one or more sensors. A sensor can be any measuring device that can be used to obtain information about it's surroundings like sonar, radar or a camera. The object or objects of interest can be for example pedestrians or aircrafts. These objects are called targets in the common engineering language. Often, the measured data is obtained periodically and one's goal is to update the current state of the target according to the data just obtained. Unfortunately, there are several sources of uncertainty e.g. some random noise could superpose with the measurement or the sensor does not detect a target at all, the number of targets in the field of view of the sensor is changing randomly. This makes object tracking a non-trivial task. However, there is a very powerful tool called Bayes' rule that eases recursive estimation despite the presence of uncertainty.

Examples of object tracking include air space monitoring, weather monitoring, cell biology and video-surveillance. This paper focuses on surveillance videos where there is a strong demand for efficient and reliable object tracking algorithms.

The goal of this thesis is to get the reader acquainted with the problematics of single and multiple target tracking. The first chapter introduces the concepts of Bayesian filtering. It presents the recursive Bayesian solution which serves as a basis for the object tracking implementations. The second chapter deals with the single-target tracking algorithms, namely the Kalman Filter and it's extended version the Extended Kalman Filter. The next chapter describes a multiple target tracking algorithm, the Nearest-Neighbour Filter. Also provides a description of a probabilistic approach to single and multiple target tracking.

The second half of this paper focuses on the implementations of the three multiple target trackers. This part contains the description of the Basic Filter that is the most simple implementation. However, this chapter provides a description of the Modified Global Nearest-Neighbour Filter and the Modified Joint Probabilistic Data Association Filter. These filters are much more sophisticated trackers than the Basic Filter. In the last chapter the three implementations are evaluated on the 2D MOT 2015 benchmark.

1 INTRODUCTION TO OBJECT TRACKING

In a typical object tracking system a number of objects (targets) move in a given region (e.g. in front of the camera) independently from each other. An object appears, stays and leaves the region randomly. The sequence of states that is followed by the target during it's lifetime is called a track. Tracking itself is basically an estimation of the target's state such as position, velocity and acceleration from noisy, corrupted and sometimes false measurements.

However it is common to make some simplifying assumptions in object tracking. This paper assumes that the targets are moving according to a Markovian process. In the systems obeying the Markov property the present state depends only on the last state and not on all previous ones. Tracking involves tracking main processes: prediction, filtering and update. This paper describes some of the different approaches implementing the filtering process. These algorithms are referred to as Filters. The phrase "filtering" stands for finding the best estimate from noisy measurements and to "filter out" the rest. [3, 13, 17]

Under the assumption that the individual targets move independently from each other, Multiple Object Tracking (MOT) breaks down into a set of Single Object Tracking (SOT) problems. Furthermore in MOT an additional problem must be addressed, because the association between measurements and targets is unknown. This problem involves making decisions that are most often based on distances. The data association methods decide which measurement to associate to a track from a cluster of measurements. This association is not always correct, there is a lot of uncertainty involved. Most of the object tracking algorithms solving these problems are based on the Bayesian approach. [13]

1.1 The Bayesian approach

In surveillance systems the measured data is typically received from frame to frame, in a sequential manner. At each stage, the last state estimate is updated according to the currently received new measurements. To handle the sequential measurements and its uncertainties the recursive form of Bayes' theorem is the suitable choice. [3]

Bayes' theorem is a tool to consistently reconcile past and current information using the conditional probability concepts of probability theory (i.e. to reason under uncertainty). Let $\boldsymbol{h}$ be the *hypothesis* and $\boldsymbol{d}$ the measurement *data* related to $\boldsymbol{h}$. According to the value of $\boldsymbol{d}$, the new state estimation is determined. Because $\boldsymbol{h}$ is a random variable and is assumed to take discrete values represented with $p(\boldsymbol{h})$, probability density function (later on pdf). The value of $p(\boldsymbol{h})$ can be defined as the

normal (Gaussian) distribution

$$p(\boldsymbol{h}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(h-\mu)^2}{2\sigma^2}}, \tag{1.1}$$

where σ is the standard deviation, σ^2 is the variance and μ is the mean of the distribution and where $\int p(\boldsymbol{h})\ \mathrm{d}h = 1$.

Having a possible outcome $\boldsymbol{h}$ and some data $\boldsymbol{d}$ that are somehow related, the *conditional distribution* $p(\boldsymbol{h}|\boldsymbol{d})$ meaning what is the probability that $\boldsymbol{h}$ happens given $\boldsymbol{d}$ is

$$p(\boldsymbol{h}|\boldsymbol{d}) = \frac{p(\boldsymbol{h}, \boldsymbol{d})p(\boldsymbol{h})}{p(\boldsymbol{d})}, \tag{1.2}$$

where $p(\boldsymbol{h}, \boldsymbol{d})$ is the joint probability of the event $\boldsymbol{h}$ and data $\boldsymbol{d}$, $p(\boldsymbol{h})$ is called the *prior distribution* and $p(\boldsymbol{h}|\boldsymbol{d})$ is called the *posterior distribution* (i.e. the new distribution of $p(\boldsymbol{h})$). Applying 1.2 twice, Bayes' theorem becomes

$$p(\boldsymbol{h}|\boldsymbol{d}) = \frac{p(\boldsymbol{d}|\boldsymbol{h})p(\boldsymbol{h})}{p(\boldsymbol{d})}, \tag{1.3}$$

where $p(\boldsymbol{d}|\boldsymbol{h})$, meaning what is the probability of $\boldsymbol{d}$ given $\boldsymbol{h}$ happens, is called the *likelihood function*. Since $p(\boldsymbol{d}|\boldsymbol{h})$ over the state-space of h is not a probability distribution ($\int p(\boldsymbol{d}|\boldsymbol{h})\ \mathrm{d}h \neq 1$) it has to be normalised with $p(\boldsymbol{d})$. The posterior distribution is obtained by re-weighting the prior distribution with the likelihood function. Normalising it with the *normalisation factor* creates a pdf from the initial value. Algorithms estimating parameters of dynamic processes are built on this concept of updating $p(\boldsymbol{h})$ with respect to $\boldsymbol{d}$ [3].

1.1.1 The recursive Bayesian solution

In object tracking the hypothesis translates to target *states* that are represented with the joint pdf $p(\mathbf{S}^k) = p(\mathbf{S}_k, \mathbf{S}_{k-1}, ..., \mathbf{S}_0) = p(\mathbf{S}_k, \mathbf{S}^{k-1})$, where $p(\mathbf{S}_k)$ is a single state at time t_k. Similarly the data translates to sensor *measurements* that are represented with $\boldsymbol{m}^k = (\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k) = (\mathbf{m}_k, \mathbf{m}^{k-1})$ where $\mathbf{m}_k$ is a single measurement at time t_k. With the substitution of $\boldsymbol{h}$ as $\mathbf{S}^k$ and $\boldsymbol{d}$ as $\mathbf{m}^k$ into equation 1.3 the fundamental equation of target tracking is obtained. However in target tracking the measurements are being obtained periodically over time and at each stage the conditional distribution $p(\mathbf{S}^k|\mathbf{m}^k)$ is updated with the latest measurement $\mathbf{m}_k$ so the Bayesian solution 1.3 must be extended to

$$p(\mathbf{S}^k|\mathbf{m}^k) = \frac{p(\mathbf{m}_k|\mathbf{S}_k)}{p(\mathbf{m}_k|\mathbf{m}^{k-1})} p(\mathbf{S}_k|\mathbf{S}_{k-1}) p(\mathbf{S}^{k-1}|\mathbf{m}^{k-1}). \tag{1.4}$$

This equation, called the recursive Bayesian solution takes into account that the joint conditional distribution $p(\mathbf{S}^k|\mathbf{m}^k)$ is being updated only by the measurement at time t_k. As mentioned above this equation is also simplified by the assumption that the process is Markovian and so the present state depends only on the last state. The likelihood function $p(\mathbf{m}_k|\mathbf{S}_k)$ says that the measurement at time t_k is independent from the prior measurements and it only depends on the current target state. The equation 1.4 is recursive because after each stage the posterior distribution is used as the prior distribution of the next stage.

Because $p(\mathbf{S}^k)$ stands for multiple states a single state must be separated out from 1.4 with integration

$$p(\mathbf{S}_k|\mathbf{m}^k) = \frac{p(\mathbf{m}_k|\mathbf{S}_k)}{p(\mathbf{m}_k|\mathbf{m}^{k-1})} \int_{\mathbf{S}_{k-1}} p(\mathbf{S}_k|\mathbf{S}_{k-1}) p(\mathbf{S}_{k-1}|\mathbf{m}^{k-1}) d\mathbf{S}_{k-1}, \tag{1.5}$$

that is the posterior pdf of the target's state at time t_k conditioned on all the measurements. The term $p(\mathbf{S}_k|\mathbf{S}_{k-1})$ is called the *transition density function* and $p(\mathbf{S}_{k-1}|\mathbf{m}^{k-1})$ is the prior pdf. The result of this integral will be the *predicted state* of $\mathbf{S}_k$ based on the previous state $\mathbf{S}_{k-1}$. Although the approaches of target tracking algorithms differ they are all trying to solve the equation 1.5. The full derivation of 1.5 and more details can be found in [3].

1.2 The optimal Bayesian filter

The aim of the optimal Bayesian filter is to recursively compute the posterior pdf using 1.5 as the base equation. In target tracking the prior distribution is represented by the previous state estimate and the likelihood function determined by the relationship between the state estimate and measurement. This filter finds the solution to 1.5 in two steps [3].

1. The transition density function is obtained from the dynamic equation of the target. The posterior pdf of the previous step is now nominated as the prior pdf. These two terms together make up the predicted density $p(\mathbf{S}_k|\mathbf{m}^{k-1})$ that is the state estimation taking into account only the information received until time t_{k-1}. The predicted density is determined by its mean value and its variance.
2. The likelihood function is obtained from the measurement t_k. After the renormalisation, together with the predicted density they result in the posterior pdf. The posterior pdf is also determined by its mean and its variance.

2 SINGLE OBJECT TRACKING

This chapter deals with some of the implementations that are based on the optimal Bayesian Filter. It introduces the Kalman Filter as most of the *MOT* algorithms are based on this filter. Also an extension of this filter is introduced that is used for estimations in non-linear systems. As their base equation for calculating the posterior distribution they use the equation 1.5 but approach it under different constraints. This chapter assumes that there is exactly one target and only one measurement obtained in each frame.

2.1 Kalman Filter

The Kalman Filter (*KF*), named after Rudolf E. Kálmán is one of the most important recursive estimators that at time t_k produces a state estimate of the target from the measurement $\mathbf{m}_k$ using Bayesian inference. Even though the measurements can contain statistical noise, it produces an estimate that is more precise than the algorithms relying solely on the measurement. The *KF* is used in navigation (GPS), in signal processing, robotic motion planning or even for smoothing the output of laptop trackpads. Furthermore the computations of the *KF* can be performed really quickly. It calculates the resulting state just in a few steps making it very fast and it's current state estimate depends only on the previous state so it is light on memory usage. [13, 5]

2.1.1 Algorithm description

The algorithm is a basic example of the recursive Bayesian filter. The *KF* is based on the physical laws of motion, known control inputs and discrete measurements. It's a two-step process:

- **state prediction:** the current state is predicted with some uncertainty
- **state update:** the estimated state is corrected with the measurement

It's important to note that the *KF* assumes that the target moves according to a linear set of equations and also the measurement is obtained by a linear equation. Secondly, the error terms of the state and sensor prediction are normal distributions.

State prediction

Assuming that the the target can move only in two dimensions it's state at time t_k will be

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \tag{2.1}$$

a four dimensional state vector where x is the position and $\dot{x}$ the velocity in the x axis, y is the position and $\dot{y}$ the velocity in the y axis. The *KF* assumes that the positions and velocities are random and normally distributed according to 1.1.

To predict the position and velocity, the basic kinematic formulas

$$\begin{aligned} x_k &= x_{k-1} + \Delta t \dot{x}_{k-1} + \frac{1}{2} a \Delta t^2, \\ y_k &= y_{k-1} + \Delta t \dot{y}_{k-1} + \frac{1}{2} a \Delta t^2, \\ \dot{x}_k &= \dot{x}_{k-1} + a\Delta t, \\ \dot{y}_k &= \dot{y}_{k-1} + a\Delta t \end{aligned} \tag{2.2}$$

are used where Δt is the elapsed time between two measurements. These linear functions can be written down in matrix forms as

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 \\ \Delta t \\ \Delta t \end{bmatrix} a \\ &= \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k \end{aligned} \tag{2.3}$$

where $\mathbf{F}_k$ is the *prediction* matrix, $\mathbf{B}_k$ is the *control* matrix and $\mathbf{u}_k$ is the control vector. The changes added by the control vector are not related to the state itself (e.g. a pedestrian suddenly starting to run is modelled as acceleration added to the state estimate). As mentioned above, every parameter of the initial state vector is considered random hence the parameters estimated by the *KF* will also be described as a Gaussian distribution. Gaussian distributions are described with their mean value and variance. The vector $\mathbf{x}_k$ describes the mean value and the covariance matrix $\mathbf{P}_k$ captures the variances, where each element is the degree of correlation between the i^{th} and j^{th} state variable. The elements on the main diagonal represent the variances

and the off-diagonal elements represent the covariances of the corresponding terms of the state vector

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{E}_{x_k}, \tag{2.4}$$

where $\mathbf{E}_{x_k}$ is the expected variance in the state.

In other words, at each prediction stage the position of the target is being estimated alongside with the uncertainty of this position. The equations 2.3 and 2.4 represent the predicted state of the target $p(\mathbf{S}_k, \mathbf{m}^{k-1})$ [14, 3].

State update

The sensor measurement received at each stage is also an estimation to some degree. The main idea of the *KF* is to average the prediction with the newly received measurements. The prediction and measurement are averaged out using a weighted average where the weights are determined by the covariance matrix calculated in the previous step. By doing so, the resulting estimate becomes the new state that is somewhere between the predicted and measured data and is the most likely new state. Because the units and scale of the measured data might not be in the same units and scale as the predicted data it has to be rescaled

$$\begin{aligned} \widehat{\mathbf{x}}_k &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k \qquad (2.5) \\ &= \mathbf{H}_k \mathbf{x}_k \\ \widehat{\mathbf{P}}_k &= \mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^T \qquad (2.6) \end{aligned}$$

where $\mathbf{H}_k$ is the transformation matrix used to map the parameters of the state vector into the domain of the measurement. This particular transformation matrix 2.5 says that even though the positions and velocities are being modelled with 2.1, when it comes to measurements it limits this model to contain only the positions in the x and y dimensions. This approach is being used when sensors measure only the position and not the velocity of the target.

Having received the measurement data $\mathbf{z}_k$ with it's uncertainty $\mathbf{R}_k$, it is averaged out with the predicted state. To calculate the average of two Gaussian distributions they have to be multiplied. The product of this multiplication is an other Gaussian distribution with it's own mean and variance. In other words the result is their

overlap, the region where both of them are likely. Mathematically,

$$\begin{aligned} \mu &= \mu_x + \frac{\Sigma_x(\mu_z - \mu_x)}{\Sigma_x + \Sigma_z}, \\ \Sigma &= \Sigma_x - \frac{\Sigma_x^2}{\Sigma_x + \Sigma_z} \end{aligned} \tag{2.7}$$

where μ_x is the mean alongside each axis and Σ_x is the covariance matrix of the estimated state, similarly μ_z and Σ_z are the mean and covariance of the measured data. From the terms above

$$\mathbf{K} = \frac{\Sigma_x}{\Sigma_x + \Sigma_z} \tag{2.8}$$

can be factored out which is called the Kalman gain. Finally, substituting 2.5 and the measurement $(\mu_z\ ,\ \Sigma_z) = (\mathbf{z}_k,\ \mathbf{R}_k)$ into 2.7 the equations for the update step are

$$\mathbf{K} = \frac{\mathbf{P}_k\mathbf{H}_k^T}{\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k} \tag{2.9}$$

$$\mathbf{x}_k' = \mathbf{x}_k + \mathbf{K}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k), \tag{2.10}$$

$$\mathbf{P}_k' = \mathbf{P}_k - \mathbf{K}\mathbf{H}_k\mathbf{P}_k. \tag{2.11}$$

The Kalman gain is basically an information measure. The larger the covariance $\mathbf{R}_k$ of the measurement the smaller will $\mathbf{K}$ be. Which means it becomes less informative. Similarly, the larger the state covariance $\mathbf{P}_k$ the larger $\mathbf{K}$ is. As 2.10 shows, three sets of information are combined to get the final state prediction. The result is acquired by adding the state estimate $\mathbf{x}_k$ to the correction $(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)$. The correction is multiplied by the gain (weighting factor) $\mathbf{K}$. A bigger value of the correction term means that the measurement and estimate are more different i.e. bigger corrections are needed. However, with a low gain, the *KF* relies more on the prediction rather than the measurement, this way the noise will be smoothed out, but the filter becomes more unresponsive. With a higher gain, the measured data are more trusted.

To get the final covariance matrix 2.11 the transformed covariance estimate weighted by the gain $\mathbf{K}$ is subtracted from the estimated covariance. Similarly to above, more information $\mathbf{K}$ has the smaller will $\mathbf{P}_k'$ be and therefore the result will be a better estimation of the final state. With zero gain the measurements are completely ignored in 2.10 and the estimated covariance is not corrected in any way.

The main disadvantage of the *KF* is that it is an estimation tool for linear models. Also the uncertainty of the model is described by a Gaussian distribution. Linear systems in practice do not exist, ultimately all of them are just approximations of non-linear systems [14].

2.2 Extended Kalman Filter

When modelling targets that move according to a linear set of equations (i.e. moving in straight lines) the *KF* is used to estimate their states. However in practice all the targets move according to a non-linear set of equation (i.e. in curved lines). The states of these models can be estimated by the Extended Kalman Filter (*EKF*). Estimating a non-linear equation is done by breaking it down to a set of linear ones. This means that the non-linear equation must be differentiated. Thus the *EKF* can be looked at as a version of the *KF* that is extended by differentiating the target's motion equations [14].

2.2.1 Algorithm description

Although this algorithm is a more advanced recursive Bayesian filter, it is only a three-step process:

- **model linearisation:** compute the Jacobian of the target's motion equations
- **state prediction:** the current state is predicted with some uncertainty
- **state update:** the estimated state is corrected with the measurement

It's important to note that the *EKF* assumes that the error terms of the state and sensor prediction are normal distributions.

Model linearisation

A target's state can be defined by a five dimensional vector

$$\mathbf{x}_k = \begin{bmatrix} x & y & \theta & s & \phi \end{bmatrix}^T \tag{2.12}$$

where x and y are the positions in x and y axis, θ is the orientation, s is the speed and ϕ is the steering angle. Assuming that the target being tracked is a robot [3] the state model can be described by

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} x_{k-1} - \Delta t v_{k-1} \sin(\theta_{k-1} + \phi_{k-1}) \\ x_{k-1} - \Delta t v_{k-1} \sin(\theta_{k-1} + \phi_{k-1}) \\ \theta_{k-1} - \Delta t v_{k-1} \tan(\phi_{k-1})/d \\ s_{k-1} \\ \phi_{k-1} \end{bmatrix} + \mathbf{v}_k \\ &= f(\mathbf{x}_{k-1}) + \mathbf{v}_k \end{aligned} \tag{2.13}$$

where f is the state-transition function and b is the distance between the wheel axis. Similarly a non-linear measurement model is described by

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{R}_k, \tag{2.14}$$

where h is the sensor function.

To make a linear approximation of a non-linear function, it has to be differentiated. In terms of target tracking these functions are differentiated with respect to the state variable $\mathbf{x}_k$ resulting in a partial derivative. A matrix of partial derivatives is called the Jacobian. The Jacobian of f and h functions are represented by

$$\mathbf{F}_k = \nabla_{\mathbf{x}_k} f(\mathbf{x}_k), \tag{2.15}$$

$$\mathbf{H}_k = \nabla_{\mathbf{x}_k} h(\mathbf{x}_k). \tag{2.16}$$

Having obtained the model equations, the latter steps of the *EKF* are similar to the *KF*.

State Prediction

The mean of the predicted state is obtained from the non-linear motion function

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{2.17}$$

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{E}_{x_k} \tag{2.18}$$

but the covariance matrix $\mathbf{P}_k$ is obtained using the Jacobian of function f.

State Update

The posterior density is obtained from the equations

$$\mathbf{K} = \frac{\mathbf{P}_k \mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k}, \tag{2.19}$$

$$\mathbf{x}'_k = \mathbf{x}_k + \mathbf{K}(\mathbf{z}_k - h(\mathbf{x}_k)), \tag{2.20}$$

$$\mathbf{P}'_k = (\mathbf{I} - \mathbf{K}\mathbf{H}_k)\mathbf{P}_k, \tag{2.21}$$

where $\mathbf{I}$ is the identity matrix. By estimating the posterior pdf from the prior pdf at each time t_k the *EKF* becomes recursive. The *EKF* is a simple method of approximating non-linear functions, however it's accuracy greatly depends on the involved uncertainty [3].

The disadvantage of this filter is that the resulting estimation is only as good as the mean of the state estimate. The bigger the non-linearity of the state model the bigger is the estimation's uncertainty. Computing the Jacobian of these functions is not always a trivial task. Furthermore, the function f may not be continuous in the range of interest [14].

3 MULTIPLE OBJECT TRACKING

Many data association methods have been developed in *MOT* systems that range from a simple nearest-neighbour assignment to a more sophisticated Multiple Hypothesis Tracker (*MHT*). The *MHT* has the best performance since it maintains hypotheses of all the possible tracks. However it has to maintain a large number of possibilities especially in clutter environments which is a NP-hard problem, since the number of possible associations is exponentially growing over k. This demands enormous computing resources. Also an *MHT* is difficult to implement. Because of these drawbacks some other algorithms were introduced that require far less computational resources yet they approximate the performance of *MHT*. However the reduction of computational complexity degrades their performance in clutter. [7]

This chapter introduces some algorithms for *MOT* that are based on the *KF*. This chapter assumes that there is zero or more targets and measurements in each frame. The most important and challenging process of *MOT* is data association. Therefore the below text focuses mainly on data association and assumes that the prediction and state update processes are the same as of the *KF*. Generally, at each time k some number of measurements are received but it is unknown which measurement belongs to which target. The task of the data association is to associate the most likely measurement (i.e. the most probable measured state) to each track from this clutter of measurements. From a single track's point of view the other measurements are considered false positive. The data association problem is especially challenging if the targets maneuver. Some amount of uncertainty is unavoidable in data association [8, 2].

3.1 Global Nearest Neighbour Filter

The Nearest Neighbour Filter (*NNF*) uses the most intuitive data association model. The basic idea of this model is to associate a single measurement $\mathbf{m}_k$ that is the closest to the predicted state estimate of the track. All the other measurements are ignored and are considered false positive.

The *NNF* that is also discussed in [8] can be interpreted as an extension of the *KF* described in 2.1 to handle multiple objects. The *NNF* becomes a four step process:

- **state prediction:** the current state is predicted with some uncertainty
- **measurement validation:** consider only the measurements that are in the gating region
- **data association:** a single measurement is selected from the validated measurements according to their distances

- **state update:** the estimated state is corrected with the selected measurement

The prediction and update step of the *NNF* is identical to the *KF*'s and will not be further described in this chapter. The data association must happen before the update step because the *KF* expects to update the predicted stage with a single measurement.

Measurement validation

Before the data association process the measurements have to be validated with a "gating" function. Common sense dictates that the next best state estimate will be near the predicted state of a track. Gating defines a region where all the measurements are considered to be associated with the track. This way all the geometrically unlikely measurements will be pruned from the obtained set of measurements. This region can be simply defined by the Mahalanobis distance as the next section shows or simply by euclidian distance.

Data association

In this step a single measurement is selected from all the received measurements. The best estimate is determined from the validated measurements by solving

$$\mathbf{z}_k = \arg \min_{\mathbf{z}_k(j), j\epsilon(1,\ldots,n)} \left[\mathbf{z}_k(j) - \mathbf{H}\mathbf{x}_k\right]^T \mathbf{N}_k^{-1} \left[\mathbf{z}_k(j) - \mathbf{H}\mathbf{x}_k\right], \tag{3.1}$$

where $\mathbf{N}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}_k$ and n is the number of measurements received at time t_k. Each element of the vector generated by 3.1 represent the Mahalanobis distance from the mean of the predicted density $\mathbf{H}\mathbf{x}_k$ to the mean of the measurements $\mathbf{z}_k(j)$. The result of the Mahalanobis distance is a measure of how many standard deviations away is $\mathbf{H}\mathbf{x}_k$ from $\mathbf{z}_k(j)$. From this distance vector the element with the lowest value is selected i.e. the measurement that is the closest to the predicted state [3]. Alternatively the closest measurement could be determined based on euclidian distances.

Applying the *NNF* in *MOT* would require to assign a *NNF* to each track. However this would produce a greedy filter, meaning the outcome of the data association process depends on the order of processing. To eliminate these drawbacks the *Global NNF* is used in case of *MOT*. The *GNNF* uses the Hungarian algorithm to find the solution to the association problem. This tracker associates a *KF* to each track, but instead of processing the tracks in a certain order the associations are determined in a single step. Section 5.2 provides a more detailed description of these filters and a comparison between a greedy and a global data association algorithm. [8]

3.2 Probabilistic Data Association

This section describes two filters that are based on the *MHT*. The *MHT* builds a tree of potential track hypotheses during the tracking. At each frame the likelihood of each track is calculated and the optimal assignment is made based on these numbers. The *MHT* considers all of the track hypotheses while tracking. This makes it a robust method but also slow and memory intensive. For this reason some filters were developed that are similar to the *MHT* yet they are computationally and memory wise much less complex. [7]

3.2.1 Probabilistic Data Association Filter

The Probabilistic Data Association Filter can be considered an *MHT* implementation that limits the tracking to a single target in clutter. This filter is also based on the *KF*. Unlike the *NNF* this filter does not select a single measurement but it combines all the measurements that might originate from the object into a single state. This means that the *PDAF* takes into account that the measurements might be erroneous. [1]

The *PDA* is also a four step process with computational requirements approximately 50% higher than those of the *KF*:

- **state prediction:** the current state is predicted with some uncertainty
- **measurement validation:** consider only the measurements that are in the gating region
- **data association:** calculate the association probabilities of the estimated state and the validated measurements
- **state update:** estimate the new state based on all the validated measurements and their association probability

Since the *PDA* is based on the *KF* the prediction equations are the same as those for the *KF* i.e. 2.3 and 2.4.

Measurement validation

The second step is to define a gating region around the estimate. The next steps will consider only the measurements that are in this region. First the innovation $\mathbf{Z}_i$ and it's covariance matrix $\mathbf{S_k}$ is calculated

$$\mathbf{Z}_i = m_i - \mathbf{x}_k \tag{3.2}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k \tag{3.3}$$

where $\mathbf{Z}_i$ is the innovation and $\mathbf{S}_k$ is it's covariance at time k. The number i refers to the i^{th} measurement. Instead of being round, the validation region is an ellipse centered at the predicted state $\mathbf{x}_k$. This shape is a product of the assumption that the error of the innovation is normally distributed.

Measurement m_i is validated by the Chi-Square test

$$\mathbf{Z}_i^T \mathbf{S}_k \mathbf{Z}_i \leq \gamma \tag{3.4}$$

where γ is a threshold corresponding to the gate probability P_G. The value of γ may be chosen from a Chi-Square table according to this probability. The gate probability P_G is the probability that the gate contains the true measurement if it is detected. Choosing an optimal value P_G plays a crucial part in gating. If it is too high the gate γ will be too large and most of the measurements will be validated and it would cause many false alarms. But if this value is too low the gate γ will also be small up to the point when just a few or no measurements can be validated. This scenario would lead to track loss. [1]

Data association

The main task of the data association process is to calculate the association probabilities β_i. First, the volume of the validation and the likelihood of the measurements are calculated, as it will be used in later computation

$$V_k = \frac{4\pi}{3} \gamma^{\frac{3}{2}} \sqrt{|\mathbf{S}_k|} \tag{3.5}$$

$$\mathcal{L}_i = \frac{N\left[m_i; \mathbf{x}_k; \mathbf{S}_k\right] P_D}{n/V_k} \tag{3.6}$$

where V_k is the volume of the validation and $\mathcal{L}_i$ is the likelihood that the i^{th} measurement is of the object and is not from clutter. Term $|\mathbf{S}_k|$ refers to the determinant of the matrix $\mathbf{S}_k$. The notation $N\left[m_i; \mathbf{x}_k; \mathbf{S}_k\right]$ represents a normal distribution centered at $\mathbf{x}_k$ with covariance $\mathbf{S}_k$, P_D is the detection probability and n is the number of the valid measurements. The denominator of 3.5 is called the spatial density (λ) which is the average rate of clutter per unit volume. Next, the probability that m_i is the correct measurement is calculated as

$$\beta_i = \begin{cases} \frac{\mathcal{L}_i}{1 - P_D P_G + \sum_{j=1}^{n} \mathcal{L}_j} & \text{if } i = 1, \ldots n \\ \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{j=1}^{n} \mathcal{L}_j} & \text{if } i = 0 \end{cases} \tag{3.7}$$

where β_i are the association probabilities and β_0 is the probability that there is no valid measurement in the gating region. While the *NNF* selects a single measurement

to update the track, the *PDAF* considers all validated measurements at state update. The *PDAF* attaches a probability to each measurement. A measurement that is more probable to originate from the object (i.e. it is close to the predicted state) has a higher association probability and vice versa. In other words these measurements play a higher role in the update process.

The state update equations use these values

$$\mathbf{v}_k = \sum_{i=1}^{n} \beta_i \mathbf{Z}_i \tag{3.8}$$

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{3.9}$$

$$\mathbf{P}_k'' = \mathbf{P}_k - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T \tag{3.10}$$

$$\mathbf{P}_k''' = \mathbf{K}_k \left[\sum_{i=1}^{n} \beta_i \mathbf{Z}_i \mathbf{Z}_i^T - \mathbf{v}_k \mathbf{v}_k^T \right] \mathbf{K}_k^T \tag{3.11}$$

where n is the number of measurements $\mathbf{v}_k$ is the combined innovation, $\mathbf{K}_k$ is the Kalman gain, $\mathbf{P}_k''$ is the covariance of the state updated with the correct measurement and $\mathbf{P}_k'''$ is the spread of the innovations term.

State update

Finally the equations for the state update are

$$\mathbf{x}_k' = \mathbf{x}_k + \mathbf{K}_k \mathbf{v}_k \tag{3.12}$$

$$\mathbf{P}_k' = \beta_0 \mathbf{P}_k + (1 - \beta_0)\mathbf{P}_k'' + \mathbf{P}_k''' \tag{3.13}$$

where $\mathbf{x}_k'$ is the updated state and $\mathbf{P}_k'$ is the covariance of the updated state. Equation 3.13 shows that predicted covariance $\mathbf{P}_k$ is weighted by β_0 the probability that any of the measurements is correct. With the probability $1 - \beta_0$ that the correct measurement is in the gating region the covariance matrix $\mathbf{P}_k''$ is updated. The final covariance of the state is also increased by $\mathbf{P}_k'''$ since it is not known which of the measurements that were validated is correct.

With these equations a single state is estimated that incorporated all valid measurements. This is a more robust approach of state update since it considers the fact that there is only one true measurement that originates from the object and the other measurements are clutter. [1]

3.2.2 Joint Probabilistic Data Association Filter

The Joint Probabilistic Data Association Filter (*JPDAF*) is one of the most widely used filters in *MOT* systems. It is an extension of the *PDAF* to multiple targets. The conventional *JPDAF* consists of four main processes

- **state prediction:** the current states of all the targets are predicted with some uncertainty
- **measurement validation:** a binary validation matrix is constructed
- **data association:** feasible event matrices are generated from the validation matrix. Each event in this matrix represents a track-to-measurement association. From these event matrices the association probabilities are calculated
- **state update:** estimate the new states of all the tracks based on all validated measurements multiplied with their association probability

This section focuses on the measurement validation and data association process, since the state prediction and update are identical with the *PDA*.

Measurement validation

The difference between the *PDAF* and the *Joint PDAF* is in the calculation of the association probabilities. Instead of calculating these values independently for each target, it determines them globally. As the first step a binary validation matrix is constructed with dimensions $m \times n$

$$\Omega = \left[\omega_{jt}\right] = \begin{pmatrix} 1 & \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ 1 & \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_{m1} & \omega_{m2} & \cdots & \omega_{mn} \end{pmatrix} \tag{3.14}$$

where n indicates the number of measurements, m the number of tracks and ω_{jt} is 1 if the measurement j is in the gating region of the track t otherwise it is 0. The value ω_{0t} implies that the measurements are from clutter. Next the feasibility matrices are calculated according to the validation matrix.

The feasibility matrices represent every possible track-to-measurement combination where every measurement j is associated only with a single track t. These associations are known as events χ and they are represented in a feasible matrix

$$\hat{\Omega}(\chi) = \left[\hat{\omega_{jt}}(\chi)\right] \tag{3.15}$$

where each value of the validation matrix Ω corresponds to the associations of event χ. The value of $\hat{\omega}_{jt}$ equals 1 only if the measurement j is from clutter ($t = 0$) or from a target t ($t \neq 0$). The number of feasibility matrices will increase exponentially with increasing m and n. [1]

Data association

For these feasible events χ the conditional probability for the association hypothesis has to be calculated according to

$$P = (\chi(\hat{\Omega} \mid \mathbf{Z}^k) = \frac{1}{c}(P_0)^{min(n,m)-m_d} \prod_{j:\omega_{jt}=1} P_{jt} \tag{3.16}$$

where $\mathbf{Z}^k$ represents all the measurements up to time k and m_d is the number of all the measurements in event χ. $\hat{\omega} = 1$ indicates that the measurement j is associated with track t. The conditional probability is normalised with c that is obtained by summing the conditional probabilities of all events. The values of P_{jt} are obtained as

$$P_{jt} = \begin{cases} N(z_j; 0; S^t)P_D & \text{if } \omega_{jt} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

where P_D is the detection probability and $N(\mathbf{z}_j; 0; S_t)$ is a normal distribution with zero mean and covariance $\mathbf{S}_t$. The last step is to calculate the association probabilities β_{jt} that will be used to update the estimated state

$$\begin{aligned} \beta_{jt} &= \sum_{\chi(\hat{\Omega})} P(\chi(\hat{\Omega} \mid \mathbf{Z}^k)\hat{\omega}_{jt} \\ \beta_{0t} &= 1 - \sum_{j=1}^{m} \beta_{jt} \end{aligned} \tag{3.18}$$

This section introduced the standard equations of *JPDAF*. Because computing the association probabilities is time expensive some new approaches had to be developed to solve the problem of time complexity. One of these algorithms was introduced by Van Wyk *et al.* [18]. This approach will be further discussed in section 5.3.3.

4 MATERIALS

This chapter briefly reviews the evaluation benchmark and the detector that was used to obtain the measurements. It also introduces two programs that were used for tracking and an other used for visualisation.

4.1 Dataset

To effectively compare algorithms of any kind a common dataset should be used. The filtering algorithms described in the next chapter were evaluated on the PETS 2009 Dataset [6]. The PETS challenge aims to detect one or more types of crowd surveillance characteristics: crowd count and density estimation, tracking pedestrians within a crowd and flow estimation.

The PETS 2009 consists of three datasets from which the S2 dataset was used which addresses pedestrian tracking. From the S2 dataset the L1 subset was used that exhibits a randomly walking sparse crowd - with a subjective difficulty level of 1. There are 5 pedestrians in each frame on average. They move with approximately the same velocity while they are in the frame. However, they may stop and continue walking in an other direction. There are also pedestrians walking in pairs. The main difficulties of this dataset arise when the pedestrians abruptly change their direction of walking and when two or more of them pass each other. Also there is an obstacle at the centre of the frame that may confuse the detector. The below figures (4.1) show some frames from this dataset.

(a) frame 250 (b) frame 500 (c) frame 750

Fig. 4.1: PETS 2009 dataset S2.L1

The S2.L1 dataset consists of an annotation and a detection file. These files consist of lines representing targets with 9 comma separated values. The annotation file contains the hand labeled ”ground truth” set of data. The first value represents the frame number and the second number is a unique ID that assigns this object to a track. Each object must be assigned to a single track. The next four values

define the object's bounding box (the box containing the object). The 7^{th} number indicates wether this entry is to be considered (1) or not (0) in the evaluation. The 8^{th} number is the type of the annotated object e.g. pedestrian (1), car (3), static person (7). The final number represents the visibility ratio of the bounding box.

The detection file of this dataset is generated by the Aggregated Channel Features detector that is discussed in the next section. The detector does not assign an ID to the objects, the second number is set to -1. The 7^{th} number denotes the confidence score of the detector. This value is indicating how confident the detector is that this measurement is a pedestrian. The last two numbers are ignored in this case. [11]

4.2 Aggregated Channel Features detector

The Aggregated Channel Features (*ACF*) detector is a straightforward and efficient detector that is often used as a pedestrian detector. The *ACF* computes several feature channels for every input image. For a colour image 10 augmented channels are calculated: 3 channels of LUV colour space, 1 channel of normalised gradient magnitude M and 6 channels of oriented gradient histogram O. Every block of pixels (e.g. 4×4) in these channels are summed and the resulting lower resolution channels are smoothed. The features are single pixel lookups the aggregated channels. Multiple rounds of bootstrapping is used to train and combine decision trees over these features. These decision trees are used to distinguish object from background. To boost detection time a rejection method is used called soft cascade. With the right choice of channels and design the *ACF* achieves state-of-art performance. Figures 4.2 show 2 frames with the bounding boxes detected by the *ACF*. [4]

(a) (b)

Fig. 4.2: Detections by the *ACF* detector

Notice that many of the pedestrians produced more than one measurements. Figure 4.2b shows a high clutter environment where most clutter measurements are generated in places of higher object density. Also the bounding boxes may be bigger than the actual object as it is shown on figure 4.2a. The *ACF* may produce false measurements of non-pedestrian objects like it is shown on figure 4.2b.

4.3 Visual implementation

Two programs were realised in this thesis. A tracking program that takes an annotated or detection file as input and produces an annotated file containing the estimated tracks. A second program was created to visualise these tracks. This section is a short manual to the user interfaces of these programs.

4.3.1 Application: Object Tracker

The main application, Object Tracker, was written in java. Using java, an object oriented programming language made the implementation of the filtering algorithms easier. Figure 4.3 shows the User Interface (UI) of the application.

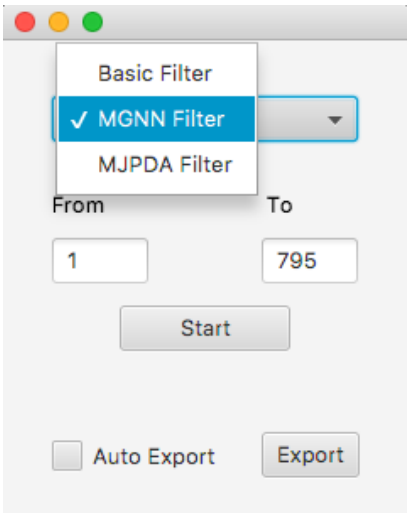


Fig. 4.3: The Object Tracker User Interface

To start the estimation process the user has to:

1. choose a filter from the listbox at the top,
2. fill in the range of estimation into the textfields *From* and *To*,
3. push the *Start* button.

After the estimation process is finished the annotation file is generated by pushing the *Export* button. Alternatively by checking *Auto Export* the annotation file is

generated automatically after the estimation process. This file is then compared to the ground truth file.

The inputs $From$ and To represent frame numbers. To successfully compare the ground truth annotation file with the estimated annotation file the number of estimated frames must equal the frame numbers in the ground truth annotation file. In this situation field $From$ must be set to 1 and field To to 795. Otherwise inputs $From$ (f) and To (t) are in range $1 \leq f, t \leq n$, where n is the number of the last frame in the dataset and $f \leq t$ must be met.

4.3.2 Visualisation Tool

This application was implemented in MatLab. It's sole purpose is to visualise the data in the ground truth and estimated annotation files. Figure 4.4 shows the UI of the application.

Fig. 4.4: Visualisation Tool User Interface

The application takes two annotation files as inputs. The application prompts the user to choose the estimated annotation file during startup. An other file can be chosen by clicking on the *Choose* button.

The ground truth and estimated tracks are shown side by side with the latter on the right. When the user presses the *Next* button the two figures will show the next frame with all the tracks drawn onto these frames up this frame. Alternatively by filling in the textfields $From$ and To and pressing the *Start* button the tracks are visualised as a video sequence.

The figures in the next chapters are generated by this tool (by pushing the *Show* button).

5 IMPLEMENTATIONS

Tracking a single object is unchallenging, given the sensor is producing a single detection of the object at every time k without any loss. However the precise location of an object is measured only by humans. As it is shown on figures 4.2 measurements received from a detector are not always accurate. The measurements can be distorted by random noise and a target can produce multiple measurements. In *SOT* these problems are addressed by filters that are able to track a single object in clutter like the *NNF* and the *PDAF*. The task of these filters is to find or produce a single best estimate from the clutter and to assign it to the track.

However in *MOT* different tracks have to be updated simultaneously. This is done by assigning a single object tracker to each track. These filters are then tracking the object to which it was assigned to independently on the other filters. The algorithm 5.1 demonstrates the basic idea behind a multiple object tracker. This algorithm updates the tracks from time $k-1$ to k.

Algorithm 5.1 Basic Multiple Object Tracker

1: $\mathbf{z}_k \leftarrow$ set of measurements at $t = k$ of size m
2: $tracks_{k-1} \leftarrow$ tracks at $t = k-1$
3: **for all** $tracks_{k-1}$ **do** ▷ $track_i \leftarrow$ the i^{th} track
4: $b \leftarrow$ FILTER$(track, \mathbf{z}_k)$ ▷ $b \leftarrow$ the best state estimate
5: **if** *ttl* of $track_i == 0$ **then**
6: *continue*
7: **end if**
8: Update $track_i$ with the best estimate b
9: **end for**
10: **if** $\mathbf{z}_k$ is not empty **then**
11: initiate new tracks
12: **end if**

Decision making in multiple object tracking in a cluttered environment is not an easy task. There is always some uncertainty in the association process. Lost tracks, track breaks or track switches are often the results of the object temporarily disappearing. For this reason all the tracks dispose with a 'time to live' (*ttl*) attribute of 5. When an object disappears temporarily the *ttl* of the track that has no measurements in it's gating region gets decremented by 1. This means that when an object disappears at time k the filter can continue to track this object up to 5 frames or time $k+5$. If the object reappears in a later frame $f \leq k+5$, the track's *ttl* gets restored to 5. When a track reaches zero *ttl* it is not updated by any other measurements.

The targets may not only disappear temporarily but also leave the frame. The *ttl* makes it possible to keep track of these targets. When a track approaches the edges of the frame the algorithm expects that this target will leave the frame. The maximum *ttl* of these tracks is set to 2. This results in a faster response when the target leaves the frame, thus generating less false positives.

When a measurement gets associated to a track it gets removed from the set $\mathbf{z}_k$. This is done to prevent other tracks to update their state with the same measurement and to control the number of tracks in the frame. Any unassociated methods will generate new tracks.

This chapter comprises of a detailed description of three filters that were implemented. All implementations are based on algorithm 5.1 when dealing with an object temporarily disappearing or leaving the frame. However their FILTER function differs, more specifically lines from 3 to 9 are replaced by the implementations.

5.1 Basic Filter Implementation

The Basic Filter (*BF*) is an implementation of the 5.1 algorithm. As it's name implies this filter serves as a basic example of filtering in multiple target systems and also as a comparison to other filter implementations. The *BF* could be considered a simpler version of the *NNF* that relies purely on the measurements. The following text discusses how does the *BF* implements the FILTER function in algorithm 5.1. The *BF* assumes that the target state is represented by a two dimensional vector

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \end{bmatrix} \tag{5.1}$$

where x and y are the positions of the target in x and y axis.

The basic idea of filtering is based on measuring distances between a known state of the track and all the obtained measurements in time k. Intuitively the track should be updated with a measurement that is close to it's last known state. The closest of these measurements is denoted best estimate and added to the track. Note that the tracks are updated in a random order as algorithm 5.1 indicates. However, when a target temporarily disappears most of the tracks are forced to pick the second or an even farther best estimate. This demands a method called gating, that validates the measurements according to their distance to the last state of the track.

First the distances between the prior state of a track and all the measurements measured. The closest measurement is then obtained by solving

$$\mathbf{z} = \arg \min_{\mathbf{z}_k(j), j \epsilon (1,\ldots,n)} \sqrt{(\mathbf{z}_k(j) - \mathbf{x}_{k-1})^2}, \tag{5.2}$$

where j is the j^{th} measurement and $\mathbf{z}$ is the closest measurement to the prior state $\mathbf{z}_{k-1}$. The closest measurement is then validated by comparing it's distance to the prior state to a constant g. If this distance is less than g the measurement is considered valid and it can be assigned to the track. The below figure 5.1 demonstrates the process of data association by the *BF*

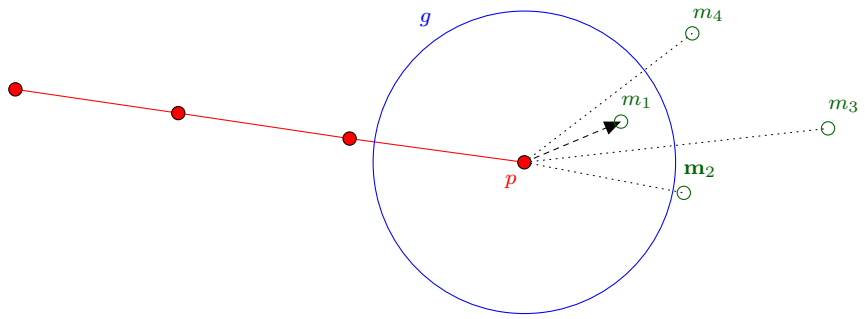


Fig. 5.1: Data association by the *BF*

where p is the prior state estimate, m_x are the measurements and g is the radius of the gating region. It demonstrates a scenario when m_1 is chosen to be the best estimate. Also all the other measurements are considered not valid. A later figure 5.5 of the same situation shows that just by defining the gating region around the predicted state the resulting association is quite different. The tracks estimated by the *BF* have a *ttl* of 1 instead of 5. This is because this filter does not produce any predictions that could be used to update the track in lack of measurements.

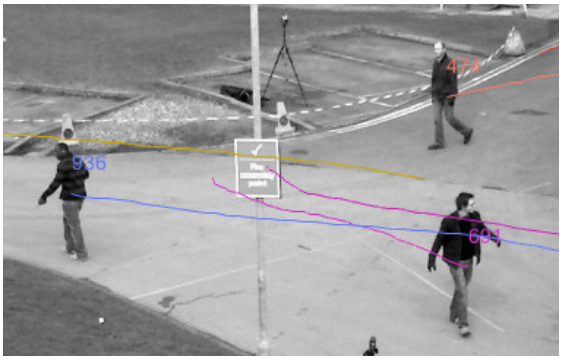
An other disadvantage of this filter lies in the presence of an order while processing the tracks. The *BF* is a greedy filter. Section 5.2.2 introduces a process called clusterisation. By organising the tracks and measurements into clusters the result of the association process will become independent on the order of processing the clusters. A comparison between the greedy *BF* and the global *GNNF* is described in detail in section 5.2.3.

The following two sections will demonstrate how this filter behaves having hand labelled and real sensor measurements. These figures were generated by the Visualisation Tool described in the previous chapter. Later sections will show the same sequences of frames to compare this and the other filters.

5.1.1 Dealing with ground truth data

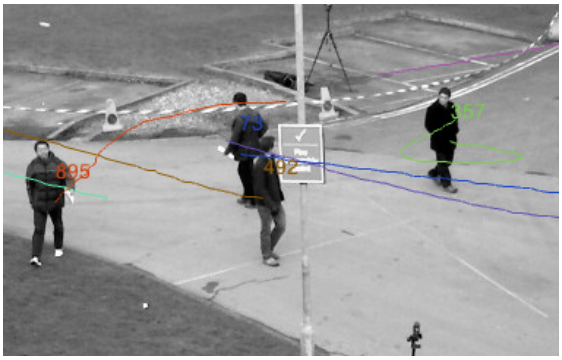
This implementation assumes that the maximum allowed distance from the prior state estimate is 35 pixels i.e. a gating region g with radius 35. In most of the sequences on figures 5.2 the *BF* could estimate the right tracks. Such high precision from the *BF* can be expected only with ground truth data as it's input. Sequence 3 (5.2c) shows that this filter is prone to wrong associations. At some point in this sequence a wrong track-to-measurement association was made. The associated measurement was deleted from the set of measurements $\mathbf{z}_k$ so the other filter associated the wrong measurement to it's track too. In this point of the frame not the global solution was found to the association problem thus resulting in track switch. This example demonstrates the disadvantages of a greedy association method. The sequences show that the *BF* is as accurate as the measurements. However in most situations when two or more targets pass each other, it fails to associate the right measurements to the tracks.

(a) Sequence 1



(b) Sequence 2

(c) Sequence 3



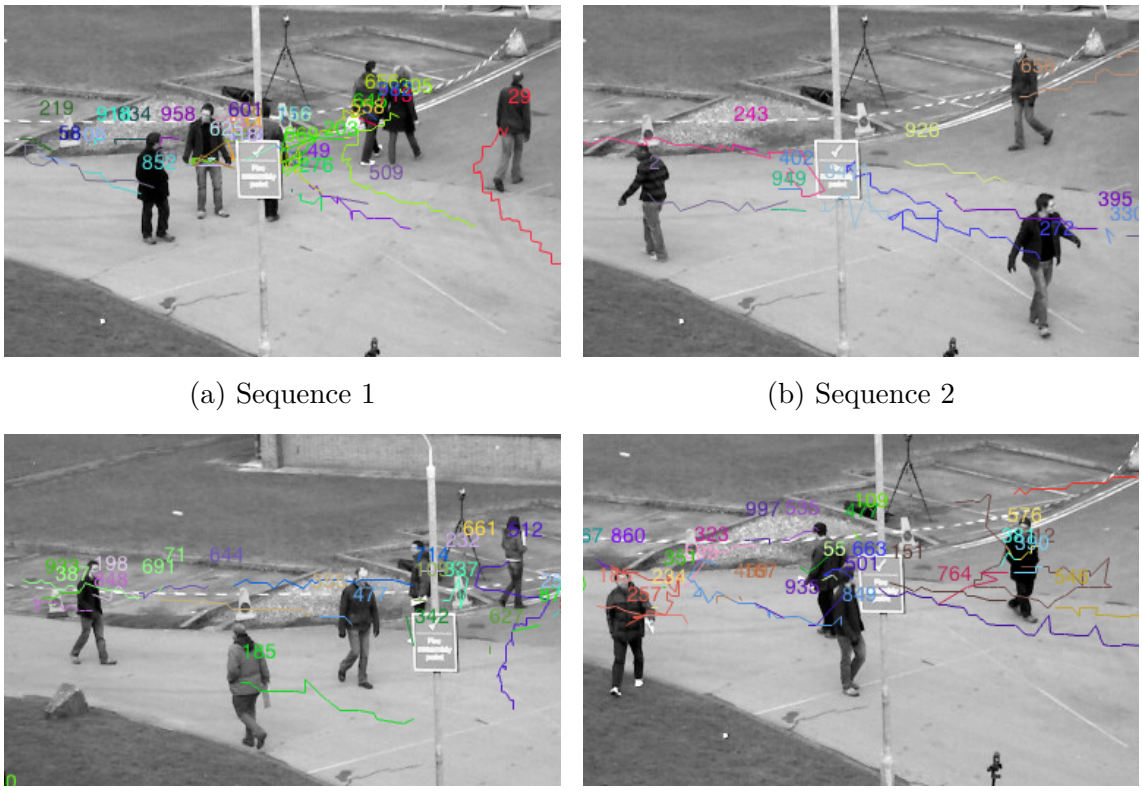
(d) Sequence 4

Fig. 5.2: The *BF* dealing with ground truth data

5.1.2 Dealing with a real sensor

Figures 5.3 demonstrate how the *BF* behaves in a high clutter environment. The measurements were not pre-processed and the outputted annotation file was not post-processed in any way. Therefore these figures also demonstrate how noisy are these measurements. The figures show that when the targets pass behind the lamp post in the middle of the frame the sensor gets confused. Noisy measurements and clutter are generated in this area. Sequence 1 (5.3a) shows that the tracks around this area are fragmented by the *BF*. All figures demonstrate that there are frames when some objects were not detected, also producing track fragmentation. The detector gets also confused when two or more objects are close to each other, thus generating clutter and resulting in fragmented tracks.

The *BF* can not distinguish real measurements from false alarms. This results in really noisy tracks as it is demonstrated on figures 5.3. They prove that relying solely on measurement data is not a good choice in object tracking. The *BF* demonstrates that, up to a certain extent, not only this but all filters are as good as the measurements on their input.



(a) Sequence 1

(b) Sequence 2

(c) Sequence 3

(d) Sequence 4

Fig. 5.3: The *BF* dealing with real measurement data

5.2 Modified Global Nearest Neighbour Filter Implementation

This implementation is based on the *NNF* introduced in section 3.1 and it was also influenced by [8]. Instead of working with the measurements this filter works with clusters that are formed of predicted states and measurements. The data association method of this algorithm is not greedy, meaning it finds the globally optimal solution to the association process. The algorithm 5.2 is a more refined implementation of the algorithm 5.1 and it replaces the lines from 3 to 9.

Algorithm 5.2 Nearest-Neighbour Filter

1: $p_k \leftarrow$ make predictions of all $tracks_{k-1}$
2: $clusters \leftarrow$ CLUSTERISATION(p_k, $\mathbf{z}_k$)
3: **for all** *clusters* **do**
4: ▷ examine each cluster by the number of tracks and measurements
5: **if** one-to-one **then**
6: Associate this measurement to the track using the *KF*
7: **end if**
8: **if** one-to-many **then**
9: Associate the closest measurement to the track using the *NNF*
10: **end if**
11: **if** many-to-many **then**
12: Associate each measurement to a track by finding the globally optimal solution using the *GNNF*
13: **end if**
14: **end for**

The *Modified GNNF* is a hybrid of three filters, namely the *Kalman Filter*, *Nearest Neighbour Filter* and the *Global Nearest Neighbour Filter*. This section describes why were all three filters used and under what circumstances is chosen one over the other.

5.2.1 State prediction

The first step of this algorithm is to calculate the predicted state of the tracks. The predicted state of a track $\mathbf{x}_k$ is the result of 2.3 and its variance $\mathbf{P}_k$ results from 2.4. The expected variance $\mathbf{E}_{x_k}$ in 2.4 is defined as

$$\mathbf{E}_{x_k} = \begin{matrix} \\ x \\ y \\ \dot{x} \\ \dot{y} \end{matrix} \begin{matrix} \begin{matrix} x & y & \dot{x} & \dot{y} \end{matrix} \\ \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \end{matrix} \tag{5.3}$$

The uncertainty or variance of the object's predicted state depends on the control input therefore $\mathbf{E}_{x_k}$ is obtained as the product of $\mathbf{B}_k^T \times \mathbf{B}_k$. The down-diagonal (or non-zero) values of $\mathbf{E}_{x_k}$ represent the variance between $\mathbf{x}_k$ and $\mathbf{B}_k$. The variance is the degree by which $\mathbf{x}_k$ changes with respect to its expected value. The up-diagonal (or zero) values represent the degree by which the input to x and y axis change with respect to each other i.e. their covariance. The covariance being zero means that they do not depend on each other.

The variance of the measurement $\mathbf{R}_{z_k}$ is defined as

$$\mathbf{R}_{z_k} = \begin{matrix} \\ x \\ y \end{matrix} \begin{matrix} \begin{matrix} x & y \end{matrix} \\ \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \end{matrix} \tag{5.4}$$

where σ_x^2 and σ_y^2 represent the variances of positions x and y. It also states that the covariance of x and y is 0 meaning they do not depend on each other.

5.2.2 Clusterisation

After predicting the new states of all the tracks the algorithm continues with a two cycle clusterisation process. During the the first cycle t clusters are formed, where t is the number of tracks currently in the frame. These clusters are represented with an estimated state of a track and zero or more measurements that passed the gating criterion. The gating region is simply represented by a circular area that is around the estimated state. In the next cycle the superclusters are formed. Superclusters are formed when two or more clusters overlap but only in a situation where there is one or more measurements in the overlapping area of these clusters. In other words these clusters are merged. The measurements that are outside of every cluster are not considered in the estimation process, instead they generate new tracks. Figure 5.4 shows some possible clusters and superclusters.

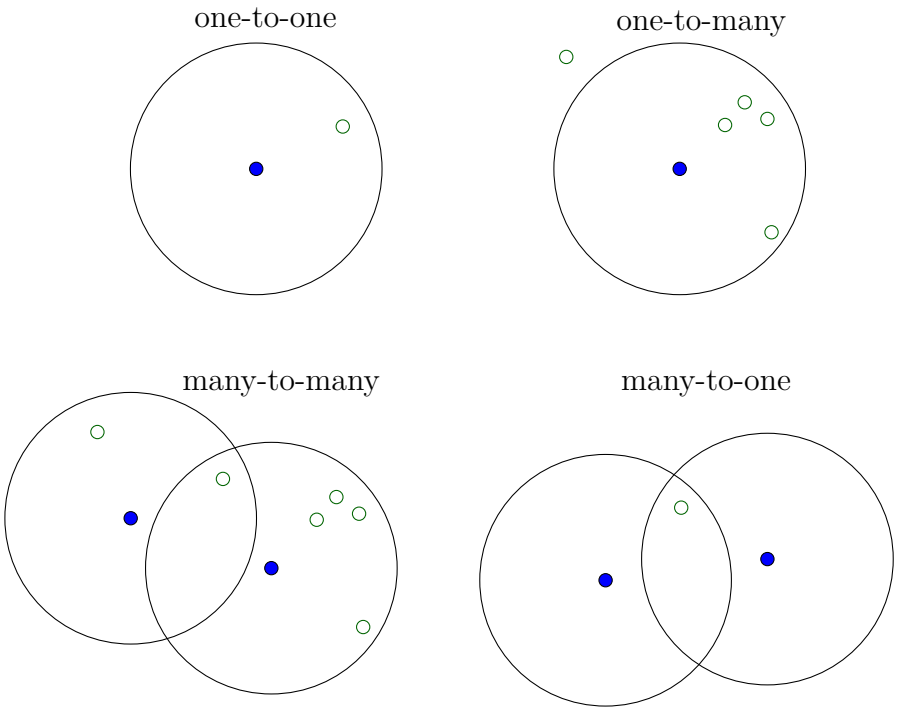


Fig. 5.4: Clusterisation by the *Modified GNNF*

Common sense dictates that the targets close to the camera are bigger that the targets further away. This means that from the camera's point of view the closer objects are travel more space when compared to an object far away even if their velocities match. Using a constant value as the radius of the clusters would mean that the closer objects could be lost because of a small radius. The clusters in the distance could be too big, leading to the creation of superclusters and the possibility of track loss. This is why the radius of the clusters are generated dynamically based on the vertical position y of the given predicted state.

5.2.3 Data association

Finding the global best estimate is called the bipartite association problem. As the result of this problem, all measurements are associated to a track in a way that the global cost is minimum. To solve this assignment problem the Hungarian algorithm was used. This algorithm has a time complexity of $O(n^3)$ where n is the maximum among the number of measurements and the number of tracks. This algorithm requires high computational resources in a highly cluttered environment. However by taking advantage of the clusters some optimalisation could be made for the filtering algorithm.

Kalman Filter: When there is only one measurement in the gating region (one-to-one) it can be assigned to the track automatically. This cluster is considered as a Single Object Tracking environment, therefore it is filtered by the *KF*.

Nearest Neighbour Filter: When there are multiple measurements in the gating region (one-to-many) the closest one is assigned to the track and the other ones are denoted clutter. This cluster is filtered by the *NNF*.

This process is demonstrated on figure 5.5 where p is the prior state estimate, $\mathbf{x}_k$ is the predicted state, $\mathbf{m}_2$ is the measurement generated by the object and g is the radius of the gating region. Because measurement m_3 is outside the gating region it is not considered in the data association process. This situation is the same as on figure 5.1 yet it resulted in a completely different association. The *NNF* chose $\mathbf{m}_2$ as the best estimate because it was the closest to the predicted state of the track. All measurements that are in the cluster are removed from the input measurements $\mathbf{z}_k$. Also this example demonstrates the main source of false positives. Consider m_3 to be of clutter. Because it is not part of any other clusters it is not removed from $\mathbf{z}_k$ so it will generate a new track at time k.

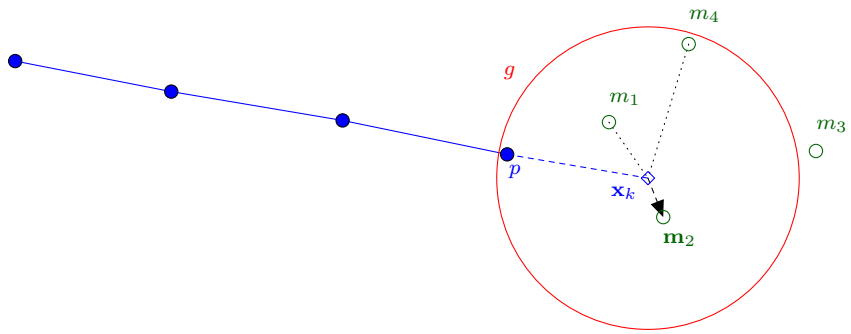


Fig. 5.5: Data association by the *NNF*

Global Nearest Neighbour Filter: Conflict situations arise when one or more measurements fall within the gating regions of two or more tracks (many-to-many). These situations are solved by the *Global NNF*'s Hungarian algorithm. In a situation is when there are less measurements than predictions in a supercluster (e.g. many-to-one), the global solution is found but the *ttl* of the tracks that have not been associated to any measurement gets decremented. Figure 5.6 demonstrates the difference between a greedy and a global association method. Note that this simple example does not consider validation. A greedy method processes the measurements in a random order so the possibility that it will find the globally optimal solution is low. A possible order is demonstrated on this figure by the numerals next to the arrows.

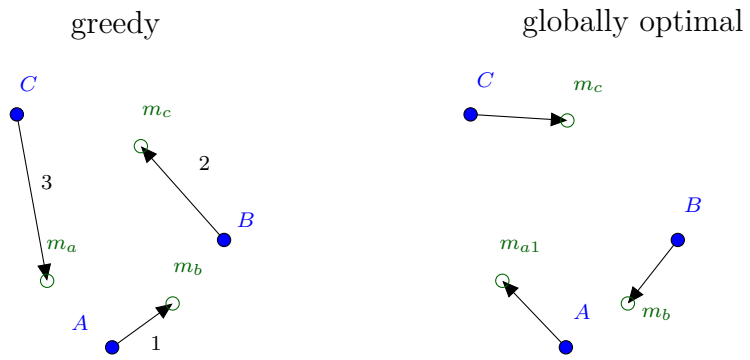


Fig. 5.6: Greedy versus global assignments

On the other hand the hungarian algorithm calculates the globally optimal solution processing the measurements as a whole. Finally all the measurements that are in the supercluster are removed from the input measurements $\mathbf{z}_k$. The example demonstrates that a greedy method of association e.g. the *BF* depends on the order of processing. The clusters are also processed in a certain order but they guarantee that no matter the order the resulting associations will be the same. With the introduction of clusters the estimation process in case of one-to-one and one-to-many clusters is reduced to a *SOT* system. Only the superclusters are considered as *MOT* systems.

In a situation when there are no measurements in a cluster, the track at it's centre loses from it's reliability so it's *ttl* gets decremented by one. This can be due to measurement loss, or the target leaving the frame. The track gets updated with the predicted state only if the track's *ttl* is greater than 0. The measurements that are not part of any clusters are generating new tracks. Because many of these measurements could be of clutter the measurements are post-processed. After the estimation process all the tracks with lengths less than 5 are omitted from the annotation file that the program generates. This reduces the amount of false positive measurements.

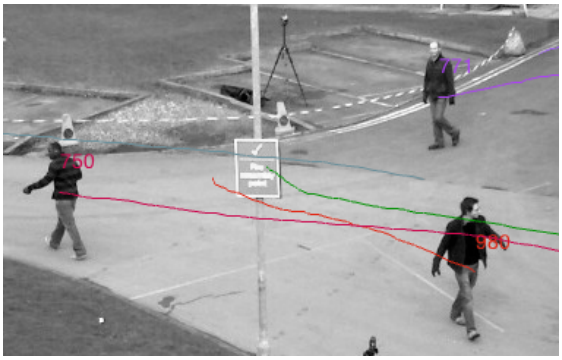
5.2.4 State update

After a cluster is processed the estimated state gets updated with the best estimate (i.e. the closest measurement). The updated state results from the equations 2.10 and 2.11. The track is updated with this state.

5.2.5 Dealing with ground truth data

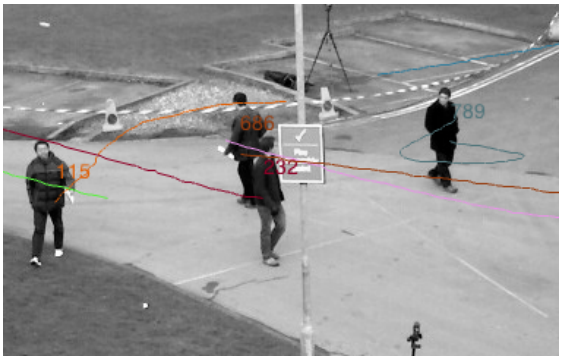
The following figures show the tracks estimated by the *Modified GNNF* where: $\Delta t = 1$, $\mathbf{u}_k = 0.002$, $\delta_x^2 = \delta_y^2 = 1$ and $ttl = 5$. The gating radius is $g = \frac{y}{10} + 5$ where y is the vertical position of a given predicted state. As figures 5.8 show, given the data is hand labelled, the *MGNNF* could estimate the right tracks in all sequences. The annotation files do not contain any clutter and are not noisy. This is why this filter performed so well. Since the estimation process involves updating the track with an estimate instead of a measurement the tracks slightly deviate from the ground truth tracks showed on figures 5.3. However unlike to the *BF* it could estimate the right tracks in sequence 3 using the hungarian algorithm.

(a) Sequence 1



(b) Sequence 2

(c) Sequence 3



(d) Sequence 4

Fig. 5.7: The *MGNNF* dealing with ground truth data

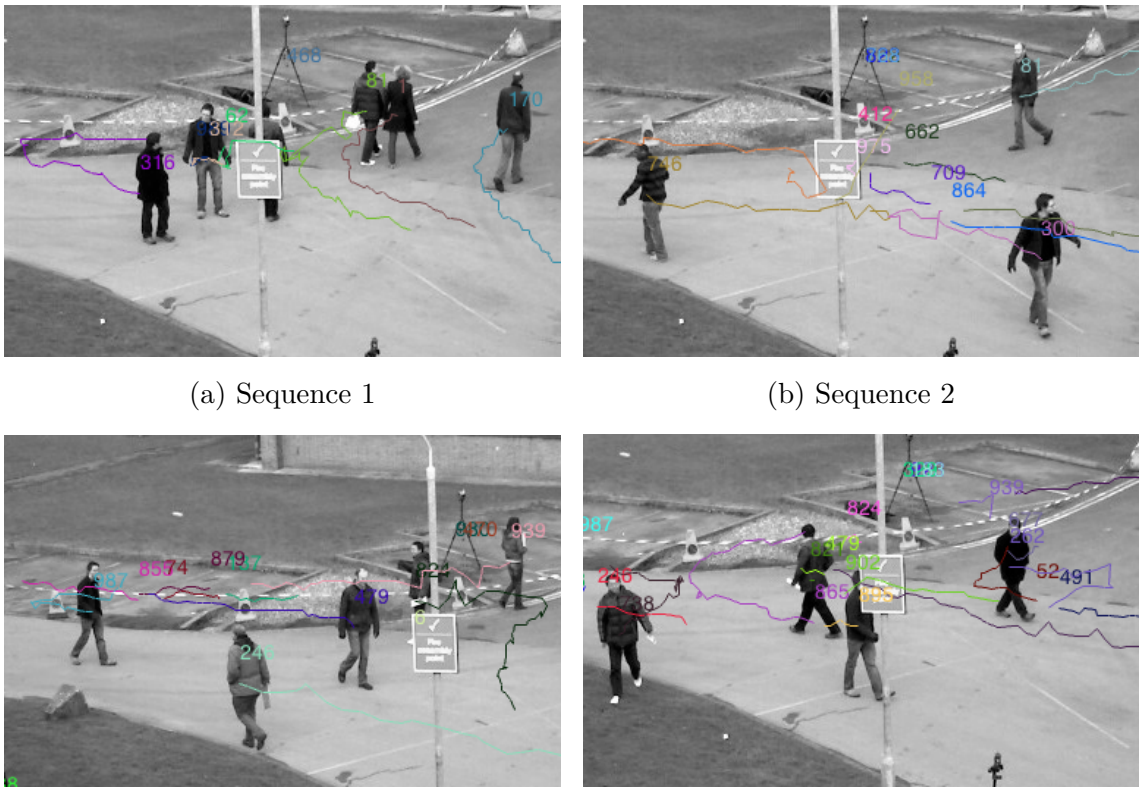
5.2.6 Dealing with a real sensor

Sequences where the measurements were obtained from a real sensor are shown in figures 5.8. In comparison with the estimated tracks on figures 5.3 the *MGNNF* generates significantly better tracks. Sequence 5.8a shows that the *MGNNF* could successfully estimate the tracks of the targets although figure 5.3a shows that this

sequence is highly cluttered. With the usage of clusters most of the clutter was filtered out, and also the right associations were made.

Sequence 5.8b shows an area around the lamp post, where the detector produced noisy measurements. As it was described before the state of an object is represented by the vector 2.1 where x and y are the centres of the object's bounding boxes. A measurement's state 5.1 is also obtained by calculating the centre of it's bounding box. The bounding boxes of the targets behind and near the lamp post are larger than of the other measurements as 4.2a shows. By calculating their centres the measurements' states may lie far from the true position of the object. In these frames the *MGNNF* may associate the wrong measurement to the track that is of clutter. These associations often result in track loss or track switches.

Sequences 5.8c and 5.8d demonstrate that the *MGNNF* is prune to track loss where the targets are densely situated. The figures show that in these places many tracks were lost and a number of new track were generated in their place.



(a) Sequence 1

(b) Sequence 2

(c) Sequence 3

(d) Sequence 4

Fig. 5.8: The *MGNNF* dealing with real measurement data

5.3 Modified Joint Probabilistic Data Association Filter Implementation

This implementation is based on the equations presented in section 3.2.1. Similarly to the *MGNNF* the *Modified JPDA* is working with clusters and it finds the globally optimal solution in a supercluster. The *MJPDA* is a hybrid of four filters, namely the *Kalman Filter*, *Nearest Neighbour Filter*, *Probabilistic Data Association Filter* and the *Joint Probabilistic Data Association Filter*. The algorithm 5.3 substitutes the lines from 3 to 9 of the 5.1 algorithm. The prediction and update steps of this filter are the same of the *MGNNF* thus they will be omitted from this section.

Algorithm 5.3 Modified Joint Probabilistic Data Association Filter

1: $p_k \leftarrow$ make predictions of all $tracks_{k-1}$
2: $clusters \leftarrow$ CLUSTERISATION(p_k, $\mathbf{z}_k$)
3: **for all** *clusters* **do**
4: ▷ examine each cluster by the number of tracks and measurements
5: **if** one-to-one **then**
6: Associate this measurement to the track using the *KF*
7: **end if**
8: **if** one-to-many **then**
9: Find the best estimate using one of: *KF*, *NNF*, *PDAF*
10: **end if**
11: **if** many-to-many **then**
12: Calculate the association probabilities with the *PJPDAF* algorithm and update each track with the *PDAF*
13: **end if**
14: **end for**

This section provides a detailed description of the *MJPDAF*. The subsections present the problems faced with the theoretical *PDA* filter. The following sections describe how was a probabilistic approach applied on the clusters.

5.3.1 Measurement validation

This section describes clusters that would be used by the standard *PDAF* then provides an adjusted clusterisation process that is used by the *MJPDAF*.

Much like in the case of the *GNNF* clusters are formed with the predictions at their centres. The clusters contain all the measurements that pass the Chi-Squared test. Those clusters that have overlapping measurements form superclusters. The below figures show the same clusters as 5.4 but the gating regions are ellipses.

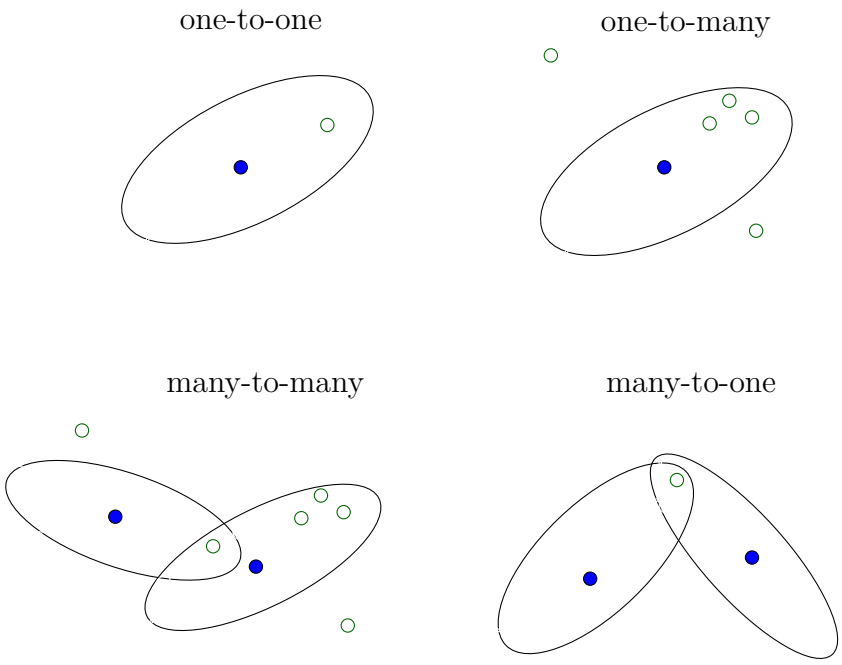


Fig. 5.9: Clusterisation by the standard *PDAF*

The shape of the standard *PDA* filter's gating region is defined by the covariance matrix 3.3. It is a gaussian distribution hence it is elliptical. This way many of the measurements are pruned that would be otherwise validated in a circular gating region. This effectively reduces the computing time of the data association algorithm. The next subsection describes why were not these clusters applicable on the detection data and how they were altered.

The optimal gating region

According to the standard *PDA* the clusters would be defined by γ. The value of γ is often chosen to be small e.g. 15. Using this value a new track was generated for every measurement in each frame. This happens because the gating region is

so small that even the closest measurements are far beyond the gating regions of all predicted states. Also the initial velocities of the targets are zero thus the first predicted state of the track is always close to the prior state. With $\gamma > 100$ the gating region is large enough that the tracks are initialised however a large gating region showed to produce a high number of assignment errors.

This problem was solved with the *KF* initialising the tracks for their first few frames and then a *PDA* algorithm could be used with $\gamma = 15$ in the rest of the frames. This value was chosen from a chi-square table to detection probability $P_D = 0,9$. The result of $\mathbf{Z}_i^T \mathbf{S}_k \mathbf{Z}_i$ for the ground truth observations that passed the chi-square test has shown to be less than 10.

This method produced good results when dealing with ground truth data, however it proved to be just slightly better than the *BF* when dealing with real measurements. The reason behind this is the smallness of γ but also the noisiness of the measurements. A small γ results in the generation of new track for every measurement that is outside the gating region. A higher value of γ is also degrading the results. Hence the *MJPDA* was adjusted by using circular clusters except for the "one-to-many" clusters where it utilises an inner elliptic too.

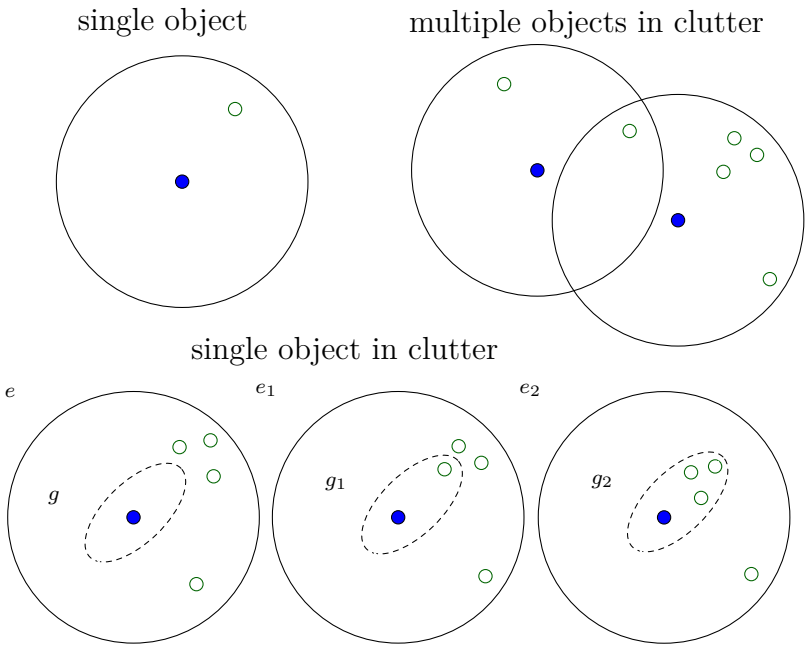


Fig. 5.10: Clusterisation by the *Modified JPDA*

5.3.2 Data association

Circular clusters eliminate the drawbacks of the *PDA* algorithm and the *MJPDA* can take advantage of them. This section describes what filters were used in the case of single object and single object tracking in clutter. Tracking multiple objects in clutter will be described in a separate section.

Single object tracking

When there is only a single measurement in the gating region the problem is reduced to single object tracking. In these situations the *PDA* algorithm would produce two values: β_1 - the association probability of the measurement - that is close to 1 and β_0 close to 0 (see equations 3.7). This cluster can be filtered by the *KF* which is faster than the *PDA* and generates the same posterior state.

Single object tracking in clutter

When there are more measurements in a cluster the problem is reduced to single object tracking in clutter. Consider a gating region g of shape $\mathbf{S}_k$ and defined by $\gamma = 50$. The *PDA* algorithm assigns a probability to each measurement in this region, then the final state estimate is calculated using all validated measurements and their corresponding probabilities. The *PDA* algorithm discussed in 3.2.1 can not be applied on a circular gating region e. The association probabilities (3.7) depend on the value of γ and expect the gating region to be elliptic. When applying equations 3.7 on the measurements inside e an association probability of 0 will be assigned to most of them. This happens because region g is smaller than e, thus there is a lower probability that the measurements will lie inside g. The measurement m, $m \in e$ but $m \notin g$ has an association probability of 0.

In cluster e a second, inner cluster g is created containing all the measurements that passed the chi-square test 3.4. Notations e and g represent the gating region of the *NNF* and of the *PDAF* respectively. One of three filters is applied on these clusters according to the number of measurements inside g. These regions are depicted on figure 5.10.

Kalman Filter: there is only a single measurement inside g. The problem is reduced to *SOT* that was described in the previous subsection. This cluster is filtered by the *KF* updating the state with the measurement inside g.

Probabilistic Data Association Filter: there are more than one measurements in the inner cluster. This is a clutter environment that can be filtered by the *PDAF* that was introduced in 3.2.1. Only the measurements inside g are used to compute the association probabilities and the posterior state.

Nearest Neighbour Filter: there are not any measurements in the inner cluster. If the *PDA* algorithm was used in this situation it would produce an association probability of 0 for every measurement and β_0 of 1. Because this would lead to track loss the *NNF* was used in it's stead that associates the closest measurement inside e to the track.

5.3.3 Projection-based Joint Probabilistic Data Association Filter

Applying the standard *JPDA* algorithm to all the measurements in every frame is time expensive. To reduce the computing time the superclusters are filtered with the Projection-based Joint Probabilistic Data Association Filter (*PJPDAF*) [18]. The standard *JPDA* algorithm has to verify many feasible association events in each frame witch requires large processing power. The *PJPDA* algorithm completely avoids the need for generating feasible matrices thus making the calculation significantly faster. The association probabilities are calculated by a Projections Onto Convex Sets (*POCS*) method. [18]

The first step is to form an $m \times n'$ matrix where each value is defined as

$$\mathbf{p}_{jt} = \begin{cases} (1 - P_D) & \text{if } t = n' \\ N(z_j; \mathbf{x}_t; \mathbf{S}_t) & \text{otherwise} \end{cases} \tag{5.5}$$

where m is the number of measurements, n is the number of tracks and n' denotes the clutter track. Also j denotes the j^{th} measurement and t the t^{th} track. The row and column constrains $\sum_{t=1}^{n'} \beta_{jt} = 1$, $\sum_{j=1}^{m} \beta_{jt,t\neq n'} \leq 1$ and $0 \leq \beta_{jt} \leq 1$ must be satisfied and $\sum_{jt}(p_{jt} - \beta_{jt})^2$ is minimal. Each value β_{jt} represents the weight of the measurement j corresponding to the track t.

When $t \neq n'$ the value p_{jt} is a scaled normal distribution with covariance $\mathbf{S}_t$ given by the innovation of the *KF* and centered at the estimated state $\mathbf{x}_t$ of target t. The row constraints when applying *POCS* to the *JPDA* optimalisation problem can be represented by the set

$$C_r = \left\{ \beta_r \in R^{mn'} : \begin{array}{c} \beta_r = \left[\beta_1, \ldots, \beta_m\right] \\ \bar{\beta_j} = \left[\beta_{j1}, \ldots, \beta_{jn'}\right] \\ \sum_{t=1}^{n'} \beta_{jt} = 1 \\ 0 \leq \beta_{jt} \leq 1 \end{array} \right\} \tag{5.6}$$

which can be shown to be closed and convex. The column constraints are represented

by the set

$$C_c = \left\{ \beta_c \in R^{m(n'-1)} : \begin{array}{c} \beta_c = \left[\beta_1, \ldots, \beta_{(n'-1)}\right] \\ \bar{\beta}_t = \left[\beta_{1t}, \ldots, \beta_{mt}\right] \\ \sum_{t=1}^{n'} \beta_{jt,t\neq n'} \leq 1 \\ 0 \leq \bar{P}_{jt} \leq 1 \end{array} \right\} \tag{5.7}$$

which is also closed and convex. The intersection of the above convex sets $C_0 = C_r \bigcap C_c$ is non-empty. Because C_0 is closed, convex and non-empty *POCS* can be used to solve the *JPDA* problem. The main idea is to calculate $T_0(\mathbf{p})$, the projection of

$$\begin{aligned} p^T &= [\bar{p}_1, \ldots, \bar{p}_m] \\ \bar{p}_j^T &= [p_{j1}, \ldots, p_{jn'}] \end{aligned} \tag{5.8}$$

onto $C_0 = C_r \bigcap C_c$. Realising a projection onto the set C_0 is complex, hence $\mathbf{T}_0$ is not directly projected. Instead T_r and T_c, the projections onto C_r and C_c were combined into a final result. The result, $\mathbf{T}_0$ is obtained by successively projecting onto C_r and C_c. The successive projection algorithm is easily implementable and the full algorithm can be found in [18]. This algorithm is executed in every time k. Each row of $\mathbf{T}_0$ represents a target t and the values represent the association probabilities of the measurements. [18]

Calculating the weights

This implementation assumes that $P_D = 0,9$, thus when $t = n'$ then the value of $\beta_{jn'} = \beta_0 = 0.1$. The equations used to obtain β_{jt} when $t \neq n'$ is

$$\beta_{jt} = \frac{e^{-\frac{1}{2}(z_j - \mathbf{x}_t)^T \mathbf{S}_t^{-1}(z_j - \mathbf{x}_t)}}{\sqrt{(2\pi)^k \left|\mathbf{S}_t\right|}} \tag{5.9}$$

where $\mathbf{S}_t$ results from 3.3 and $|\mathbf{S}_t|$ is the determinant of $\mathbf{S}_t$ and k is the length of vector $\mathbf{x}_t$. The values of β_{jt} are small even when the measurements and the track are close. The below example shows a possible matrix $\mathbf{p}_{jt}$ where the number of measurements is 3 and the number of tracks is 2 with the resulting matrix $\mathbf{T}_0$ is

$$\mathbf{p}_{jt} = \begin{pmatrix} 2.10^{-10} & 0 & 3.10^{-14} & 0.1 \\ 0 & 4.10^{-8} & 3.10^{-3} & 0.1 \end{pmatrix},$$

$$\mathbf{T}_0 = \begin{pmatrix} 0.225 & 0.225 & 0.225 & 0.325 \\ 0.224 & 0.224 & 0.227 & 0.324 \end{pmatrix}$$

This example demonstrates that small numbers (e.g. 2.10^{-10}) have little if any effect on the resulting matrix. To meet the condition $\sum_{t=1}^{n'} \beta_{jt} = 1$ each value in $\mathbf{p}_{jt}$ has to be normalised according to

$$\beta'_{jt} = \frac{\beta_{jt}}{\sum_{t=1}^{n} \beta_{jt}}, \tag{5.10}$$

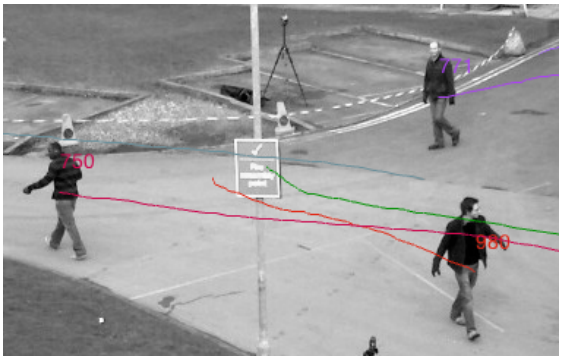
$$\beta''_{jt} = \beta'_{jt} - \beta'_{jt}(1 - P_D) = \beta'_{jt} P_D. \tag{5.11}$$

where β''_{jt} is the normalised weight. First all values in a row are divided by the sum of the t^{th} row in $\mathbf{p}_{jt}$ without including $t = n'$, se equation 5.10. This ensures that $\sum_{t=1}^{n} \beta'_{jt} = 1$ is met for each row. To meet the row constraint of $\mathbf{p}_{jt}$ each value has to be multiplied by P_D. This ensures that $\sum_{t=1}^{n'} \beta''_{jt} = 1$ is met for each row.

5.3.4 Basic Behaviour

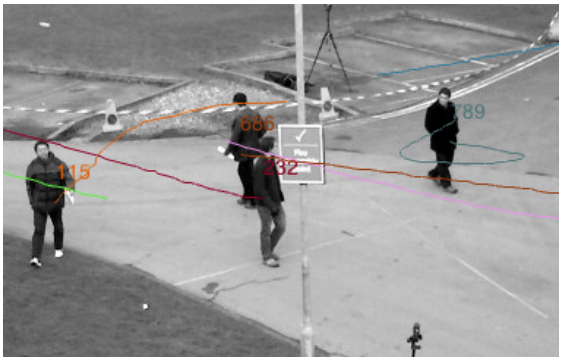
The following figures show the resulting tracks estimated by the *Modified JPDAF* where: $\Delta t = 1$, $\mathbf{u}_k = 0.002$, $\delta_x^2 = \delta_y^2 = 1$, $ttl = 5$, $P_D = 0,9$ and $\gamma = 50$. The gating radius $e = \frac{y}{10} + 5$ where y is the vertical position of a given predicted state. Similarly to the *MGNNF* the *JPDAF* estimated the right tracks of the targets given the data is hand labelled.

(a) Sequence 1



(b) Sequence 2

(c) Sequence 3

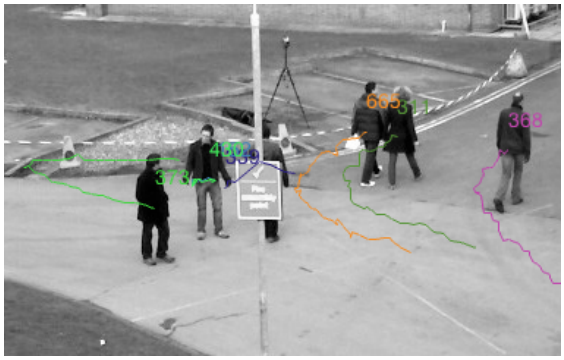


(d) Sequence 4

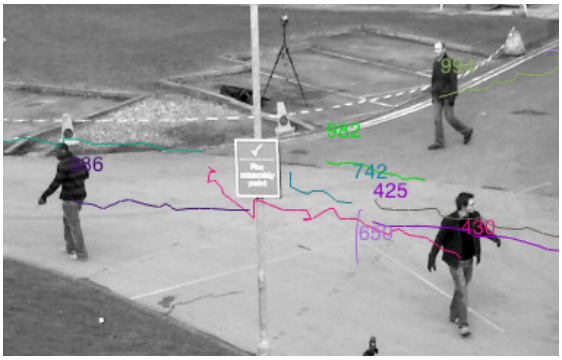
Fig. 5.11: The *MJPDAF* dealing with ground truth data

5.3.5 Dealing with a real sensor

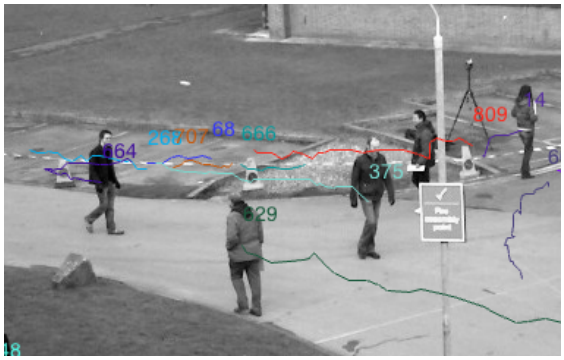
Figures 6.4 shows sequences $1-4$ where the tracks were estimated by the *MJPDAF*. The figures show a slightly better performance than the *MGNNF*. There is a slight difference between the two filters specifically in the handling of high clutter because most clusters are filtered by the *NNF*. This is why the *MJPDAF* tends to loose the tracks when two or more targets get close each other. This is demonstrated on the left side of figure 5.12d where targets 220 and 440 crossed each others paths. Both tracks were lost but the tracks were successfully recovered resulting in tracks 42 and 605 respectively. Figure 5.12b shows a similar situation however one of the targets was recovered only far from the time of collision. This resulted in a long gap in this track, but unlike the *MGNNF* that lost both tracks (see fig. 5.8b) the other target was successfully tracked.



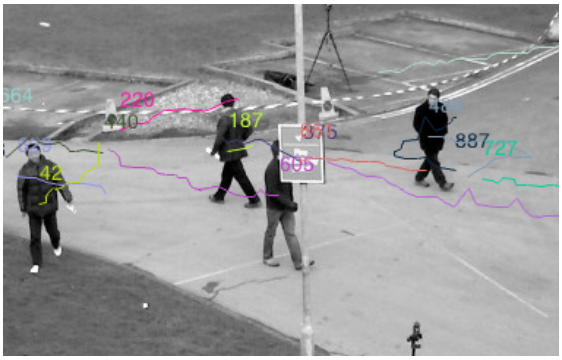
(a) Sequence 1



(b) Sequence 2



(c) Sequence 3



(d) Sequence 4

Fig. 5.12: The *MJPDAF* dealing with sensor data

6 PERFORMANCE VALIDATION

A number of techniques have been proposed for target tracking performance evaluation as [9] suggests. In a typical setup a bounding box of the target is obtained from a sensor and this is compared to the ground truth bounding box. A basic algorithm takes these boxes calculates their centres and evaluates their difference according to a threshold [15]. Another approach is to determine how many frames has the algorithm been able to follow the target [10]. In [12] a new tracking accuracy measurement was proposed which is based on how two bounding boxes overlap. To compare two statistical models an F-test could be used [16].

As mentioned before the implemented filters were evaluated on the PETS 2009 benchmark, namely on sequence S2.L1. The tracking results are based on the widely used evaluation metrics 2D MOT 2015 [11]. With a common evaluation method and by using the same ground truth data for all datasets the comparison between all the results is guaranteed to be fair.

The two basic features of a tracking algorithm is how many of the measurements were true positive (TP) that is a real target and how many were false positive (FP). False positives are determined by a distance measure from the ground truth target. A target that is missed while tracking is a false negative (FN). The result of a good tracking algorithm has to have as few FPs and FNs as possible. An ID switch (IDSW) happens when at time k a ground truth target b gets assigned to the track t but at time $k-1$ a target $a \neq b$ was assigned. A track fragmentation occurs when a ground truth target is no longer tracked but at a later point it is tracked again. A track fragmentation infers also an ID switch [11]. An example of an ID switch and track fragmentation can be seen on figure 6.1. Notation c refers to a clutter measurement and p is a predicted state. This particular fragmentation happened because at a certain time a clutter measurement c got assigned to track t_a and the true track was lost. Instead a new track t_b was generated to track the target.

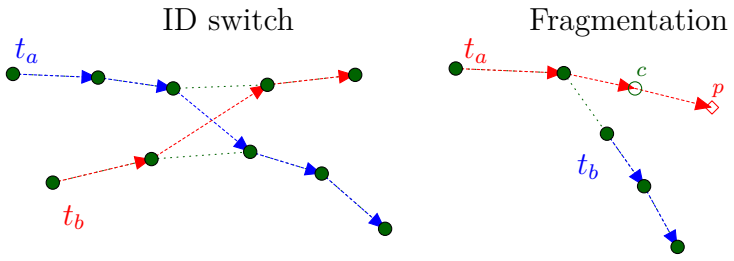


Fig. 6.1: ID switch and track fragmentation

One of most often used metrics to evaluate a tracker's performance is the Multiple

Object Tracking Accuracy (MOTA). It is defined as

$$MOTA = 1 - \frac{\sum_k (FN_k + FP_k + IDSW_k)}{\sum_k GT_k} \tag{6.1}$$

where k is the time interval or frame number and GT is the number of ground truth objects. The value of MOTA is in range $[-\infty, 100]$. A good tracker should generate tracks that have high score of MOTA.

An other metric is used to qualify the localisation capabilities of the tracker. This metric is called the Multiple Object Tracking Precision (MOTP). The precision is defined as how much do the estimated states overlap with their corresponding ground truth state.

$$MOTP = \frac{\sum_{k,i} d_{k,i}}{\sum_k c_k} \tag{6.2}$$

where c_k is the number of matches in time k and $d_{k,i}$ is the bounding box overlap of target i with it's ground truth object. This information however does not reflect the tracker's actual performance.

The quality of the tracks are also evaluated. The tracks are classified as mostly tracked (MT), partially tracked (PT) and mostly lost (ML). A track is mostly tracked when it is successfully tracked at least for the 80% of it's lifetime. On the other hand if less than 20% of the track is estimated correctly it is mostly lost. These metrics do not take into account wether there were ID switches taking place in the track. The track fragmentation (FM) is a measure of how many times was the track interrupted during it's lifetime. Evidently a high count of MT and low amount of PT, ML and FM are defining a good tracker. [11]

6.1 A comparison between the *Modified JPDAF* and the *Modified GNNF*

This section will describe a sequence that demonstrates some of the differences between the two implementations. The track estimates are shown on figures 6.2a and 6.2b. Figure 6.2c shows the true tracks of the objects. Track 762 estimated by the *MJPDAF* and track 980 estimated by the *MGNNF* are the same. This suggests that the clusters of these tracks were filtered with the *NNF*. Because the inner gating region of the *MJPDAF* is small, even with $\gamma = 50$, most measurements lie in the outer gating region. This means that in most situations filtering these clusters with the *MJPDAF* is prone to make the same errors as the *MGNNF*. However the superclusters are filtered by the *PJPDAF* or the *GNNF* respectively. Hence these are the clusters that will be always filtered with these algorithms.

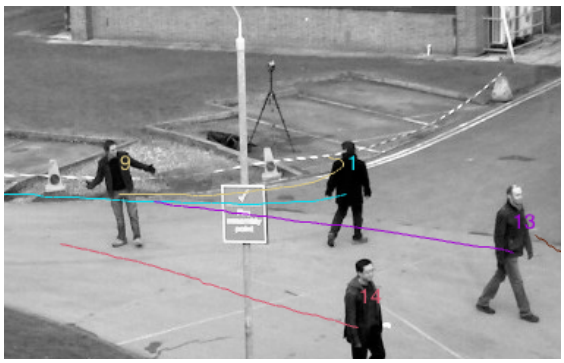
Ground truth tracks 1 and 13 are moving from left to right, and track 9 is moving in the opposite direction. These targets meet at a point behind the lamp post. At this location of the frame the sensor produces noisy measurements and a high number of clutter.

Track 74 is an estimation of track 13 by the *MGNNF*. Figure 6.2b shows that when the target was passing behind the lamp post it was lost. At this time a new track 680 was generated. However this track is a sequence of false positives (FPs). Track 13 is recovered only later on by track 657. Note that the tacks' identification numbers are random unique numbers. Ground truth tracks 1 and 9 are also lost when they pass each other behind the lamp post. Tracks 185 and 707 are the estimates of track 1, and tracks 379 and 168 are the estimates of track 9.

On the other hand, the *MJPDAF* successfully estimated two of three tracks. Track 359 is the estimation of track 13 and track 496 is the estimation of track 1. The *MJPDAF* did not loose target 13 and no new track was generated. This allowed the filter to continue the tracking of the right target. However similarly to the *MGNNF* track 9 was not successfully estimated.

(a) *MJPDAF* track estimates

(b) *MGNNF* track estimates



(c) Ground truth tracks

Fig. 6.2: A comparison between the *MGNNF* and *MJPDAF*

6.2 Benchmark evaluation

As mentioned before the filters are evaluated on the 2D MOT 2015 benchmark. Benchmark PETS2009 contains 19 ground truth tracks (GT = 19). Table 6.1 shows the results with the hand labelled data as input. Note that the results in table 6.1 are produced by the filters while using clusters with a constant radius of 20. Clusters with dynamic radii produce some FNs and FPs while filtering hand labelled data, hence they are applied only on the measurement data. The three filters perform well on the hand labelled data, all 19 tracks were mostly tracked (MT = 19). The *BF* has no false positive measurements since the tracks have a *ttl* of 1. Also it's precision MOTP = 100 since it relies solely on the data. The 17 false positives (FP = 17) by the *MGNNF* and *MJPDAF* are generated when the objects leave the frame. These filters missed 5 measurements (FN = 5), but there are no ID switches or track fragmentations. An ID switch made by the *BF* can be seen on figure 5.2c. The precision of the *MGNNF* and *MJPDAF* is less than that of the *BF*, because the states generated by these filters are indeed estimates.

Tab. 6.1: 2D MOT2015 Challenge results (ground truth)

Filter	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP
BF	19	19	0	0	0	0	3	0	99,9	100
MGNNF	19	19	0	0	17	5	0	0	99,5	98,5
MJPDAF	19	19	0	0	17	5	0	0	99,5	99,5

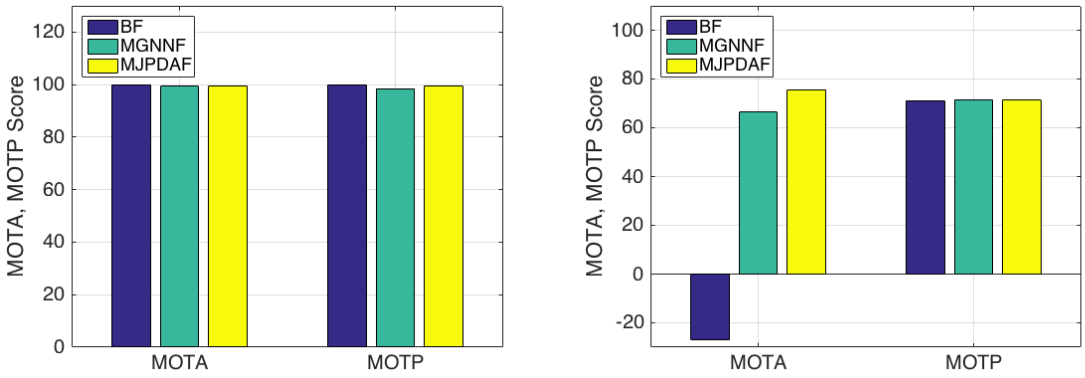
Table 6.2 shows the results with the measurement data as input. The *BF* has the lowest score of MOTA with a negative value. It generated many false positives and missed many targets. Also it has the highest amount of ID switches.

Tab. 6.2: 2D MOT2015 Challenge results (measurements)

Filter	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP
BF	19	18	1	0	1327	399	4183	138	–27,1	71,0
MGNNF	19	17	2	0	984	494	83	138	66,4	71,5
MJPDAF	19	18	1	0	516	535	85	138	75,6	71,5

This table proves that the *MGNNF* and the *MJPDAF* are significantly better filters than the *BF*. However there is only a slight improvement in the *MJPDAF*

over the *MGNNF*. It generates far less false positives but has a slight increase in false negatives and ID switches. This suggests that in most cases the *MJPDAF* degrades into *MGNNF*. Some of the FN can be attributed to the post-processing of the estimated annotation file. By deleting a track that has a length less that 5, in some cases the true position of the measurement is deleted. But this step significantly reduces the amount of FP. The figures below (6.3) depict the MOTA and MOTP scores of all filters when dealing with ground truth and measurement data.



(a) Ground truth data

(b) Measurement data

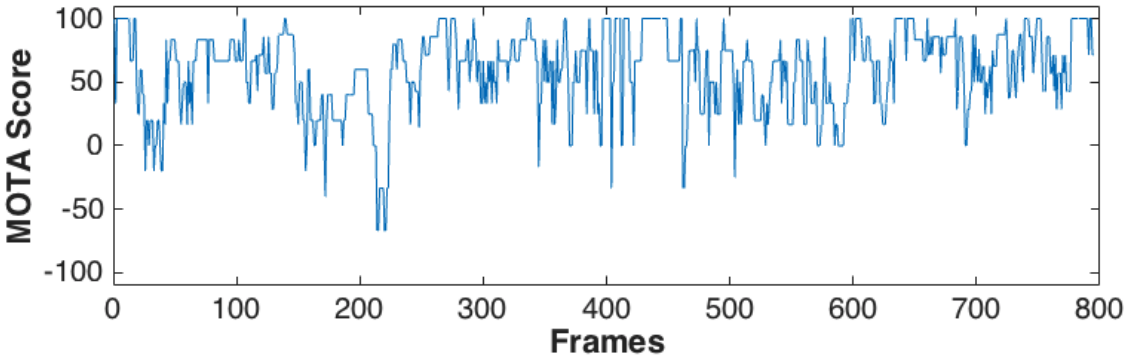
Fig. 6.3: The *GNNF* dealing with real measurement data

Most clusters are filtered by the *NNF* even if the *MJPDAF* is used, as it was explained in section 5.3.2. Therefore the increase of it's performance is due to fact that the *PDAF* is used to filter some of the clusters and to the *PJPDAF* filtering the superclusters. Some of the mistakes can be attributed to the dynamic radius of the clusters. Even though it reduces the amount of FPs the problem of this approach is that it is not universal. It means that a different dataset or a different angle of the same area will need a different equation defining the radius. These equations could then be devised through trial and error.

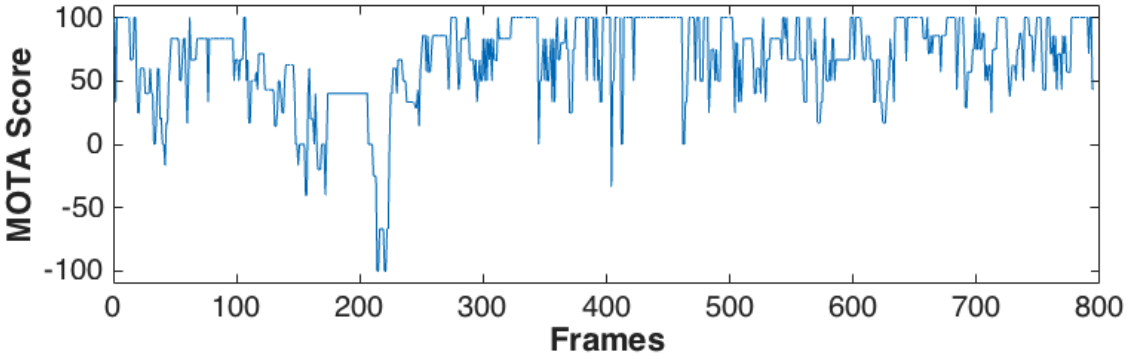
The implementations make wrong associations in areas of high density. In these situations the detector generates a high amount of FPs, resulting in wrong associations by the trackers. Without the knowledge of the targets' velocities the radius of the clusters can not be accurately modelled. This paper assumes that the targets move with a constant velocity, which may also lead to wrong associations. Therefore the results suggest that a tracker based solely on positional measurements will always be dependant on the input measurements.

Figures 6.4a and 6.4b depict the MOTA scores in each frame that were achieved by the *MGNNF* and *MJPDAF* respectively. The MOTA scores in table 6.2 are than calculated as the average of the values in 6.4a and 6.4b. These figures provide an

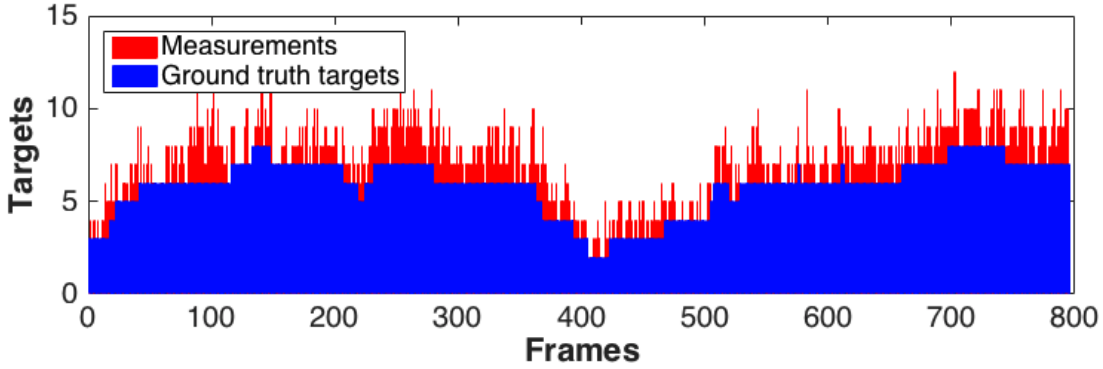
in-depth look at where did the *MJPDAF* outperform the *MGNNF* and vice-versa. Figure 6.4c provides the information on how many targets were actually in the frame (blue) and how many measurements were generated by the sensor (red) in each frame. This figure shows that in most frames the sensor generated a higher amount of measurements than there were targets.



(a) MOTA scores of the *MGNNF* in each frame



(b) MOTA scores of the *MJPDAF* in each frame



(c) Target counts

Fig. 6.4: The *GNNF* dealing with real measurement data

7 CONCLUSION

This thesis focused on multiple object tracking in a lower density environment with high clutter. The tracking is based on the knowledge of the positions of the objects in each frame. Two trackers were implemented in this thesis, namely the *Modified GNNF* and the *Modified JPDAF*.

The first part of this thesis provided a description of some of the existing single object trackers and multiple object trackers. The implemented trackers are based on these filters. The second part of this thesis described the implemented trackers in detail. The last chapter evaluated the implemented trackers and provided a comparison between the three implementations.

Amongst the implemented algorithms is a Basic Filter that is a realisation of the basic idea of multiple target tracking. This filter associates the closest measurement in a given frame to the track. This primitive behaviour has it's drawbacks. This filter performs well only when the measurements on it's input are hand labelled. Table 6.2 proves that associating tracks to measurements based only on distances does not produce good results.

The *Modified GNNF* is not a single filter but rather a set of filters. The *GNNF* is often chosen to solve the bipartite association problem, however, it has a time complexity of $O(n^3)$ where n is the maximum among the measurements and the targets this filter is used only when it is necessary. As it was mentioned above the *MGNNF* is a hybrid of three filters, namely the *KF*, *NNF* and the *GNNF*. To decide on which filter to use to update a track a method called clusterisation was introduced. A filter is then chosen to update a track based on how many measurements are in a cluster. Only a supercluster that comprises of more than one clusters is filtered by the *GNNF*. This has shown a great decrease in time complexity of these filters.

Similarly to the *MGNNF* the *Modified JPDAF* is a hybrid of five filters, namely the *KF*, *NNF*, *PDAF* and the *PJPDAF*. This filter does also take advantage of clusters, thus reducing time cost.

These filters have shown to produce good results when filtering real measurements as opposed to the *BF*. However, the *MJPDAF* shows only a slight improvement as opposed to the *MGNNF*. This is due to the fact that most clusters are filtered by the *NNF* in both the *MGNNF* and the *MJPDAF*. This results in the *MJPDAF* degrading to the *MGNNF*. However, the increase in the *MJPDAF*'s performance can be associated to the events when the clusters are filtered by the *PDA* algorithm that is designed to track a single target in clutter.

BIBLIOGRAPHY

[1] Bar-Shalom, Y., Daum, F & Huang J. (2009). The probabilistic data association filter. *IEEE Control Systems 29(6)*, pp. 82-100. doi:10.1109/MCS.2009.934469

[2] Benfold, B., & Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. *Cvpr 2011*, pp. 3457-3464. doi:10.1109/cvpr.2011.5995667

[3] Challa, S. (2011). *Fundamentals of object tracking.* Cambridge: Cambridge University Press.

[4] Dollar, P., Appel, R., Belongie, S., & Perona, P. (2014). Fast Feature Pyramids for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(8)*, pp. 1532-1545. doi:10.1109/tpami.2014.2300479

[5] Faragher, R. (2012). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]. *IEEE Signal Processing Magazine, 29(5)*, pp. 128-132. doi:10.1109/msp.2012.2203621

[6] Ferryman, J. & Shahrokni, A. (2009) PETS2009: Dataset and challenge. *Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Snowbird, UT, pp. 1-6. doi: 10.1109/PETS-WINTER.2009.5399556

[7] Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple Hypothesis Tracking Revisited. *2015 IEEE International Conference on Computer Vision (ICCV).* doi:10.1109/iccv.2015.533

[8] Konstantinova, P. and Udvarev, A. & Semerdjiev, T. (2003). A Study of a Target Tracking Algorithm Using Global Nearest Neighbor *Approach Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*, pp. 290-295, doi:10.1145/973620.973668

[9] Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay G., Porikli F. & Cehovin, L. (2016). A Novel Performance Evaluation Methodology for Single-Target Trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(11)*, pp. 2137-2155. doi:10.1109/tpami.2016.2516982

[10] Kwon, J., & Lee, K. M. (2009). Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling. *2009 IEEE Conference on Computer Vision and Pattern Recognition.* doi:10.1109/cvprw.2009.5206502

[11] Leal-Taixé, L., Milan, A., Reid, I., Roth, S. & Schindler, K. (2015). MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking

[12] Li, H., Shen, C., & Shi, Q. (2011). Real-time visual tracking using compressive sensing. *Cvpr 2011.* doi:10.1109/cvpr.2011.5995483

[13] Oh S., Russell, S. & Sastry, S. (2009) Markov Chain Monte Carlo Data Association for Multi-Target Tracking. *IEEE Transactions on Automatic Control 54(3)*, pp. 481-497, doi: 10.1109/TAC.2009.2012975

[14] Risabero, M. I. (2004). Kalman and Extended Kalman Filters: Concept, Derivation and Properties. Technical report., Institute for Systems and Robotics, Lisboa

[15] Ross, D. A., Lim, J., Lin, R., & Yang, M. (2007). Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision, 77(1-3)*, pp. 125-141. doi:10.1007/s11263-007-0075-7

[16] Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., & Shan, M. (2014). Visual Tracking: An Experimental Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7)*, pp. 1442-1468. doi:10.1109/tpami.2013.230

[17] Yang, B., & Nevatia, R. (2012). Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. *2012 IEEE Conference on Computer Vision and Pattern Recognition.* doi:10.1109/cvpr.2012.6247892

[18] Wyk, B. V., Wyk, M. V., & Noel, G. (n.d.). A projection-based joint probabilistic data association algorithm. *2004 IEEE Africon. 7th Africon Conference in Africa (IEEE Cat. No.04CH37590).* doi:10.1109/africon.2004.1406621

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

ACF Aggregated Channel Features

BF Basic Filter

EKF Extended Kalman Filter

FN False Negative

FP False Positive

FM Fragmentation

GNNF Global Nearest-Neighbour Filter

GT Ground Truth

IDSW Identification Switch

JPDAF Joint Probability Data Association Filter

KF Kalman Filter

MGNNF Modified Global Nearest-Neighbour Filter

MJPDAF Modified Joint Probability Data Association Filter

ML Mostly Lost

MT Mostly Tracked

MHT Multiple Hypothesis Tracker

MOT Multiple Object Tracking

MOTA Multiple Object Tracking Accuracy

MOTP Multiple Object Tracking Precision

NNF Nearest-Neighbour Filter

PT Partially Tracked

PDA Probabilistic Data Association

PDAF Probabilistic Data Association Filter

PJPDAF Projection-based Joint Probability Data Association Filter

POCS Projections Onto Convex Sets

SOT Single Object Tracking

TN True Negative

UI User Interface